VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zpracování signálu pomocí jádra ARM Cortex-M0+ s matematickým koprocesorem

Signal Processing with ARM Cortex-M0+ Core Including Mathematical Coprocessor

2016 Martina Kysučanová

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání bakalářské práce

Student: **Martina Kysučanová**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2601R013 Telekomunikační technika

Téma: Zpracování signálu pomocí jádra ARM Cortex-M0+ s matematickým koprocesorem
Signal Processing with ARM Cortex-M0+ Core Including Mathematical Coprocessor

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Natudování, jak pracuje jádro ARM Cortex-M0+ a jaké jsou jeho výhody oproti jiným jádrům ARM.
2. Seznámení se s procesorem řady Kinetis M.
3. Převod vstupního spojitého signálu na digitální.
4. Nastudování a implementace základních algoritmů pro zpracování jednorozměrného digitálního signálu s a bez využití matematického koprocesoru.
5. Aplikace implementovaných algoritmů na měřený vstupní signál.
6. Zhodnocení dosažených výsledků.

Seznam doporučené odborné literatury:

[1] YIU, Joseph. The definitive guide to the ARM CORTEX-M0. Oxford: Elsevier ;, c2011, xix, 529 s. Technology/electronic/microcontrollers. ISBN 978-0-12-385477-3.
[2] VALVANO, Jonathan W. Embedded systems: Introduction to the Arm Cortex(TM)-M3 microcontrollers. 2nd ed. s.l.: CreateSpace, 2012, xii, 462 s. Technology/electronic/microcontrollers. ISBN 978-1477508992.
[3] Freescale Kinetis M Reference manual.
[4] KUO, Sen M, Bob H LEE a Wenshun TIAN. Real-time digital signal processing: fundamentals, implementations and applications. 3rd ed. Chichester, West Sussex: Wiley, 2013, xvii, 544 p. ISBN 978-1118414323.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jan Tomeček**

Datum zadání: 01.09.2015

Datum odevzdání: 29.04.2016

doc. Ing. Miroslav Vozňák, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně. Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

V Ostravě 29. dubna 2016 .....

Ráda bych na tomto místě poděkovala vedoucímu bakalářské práce Ing. Janu Tomečkovi za cenné rady a odbornou pomoc při zpracování bakalářské práce.

Abstrakt

Tato práce se zaměřuje na práci s jádrem ARM Cortex M0+, jeho použitím a využitelností oproti jiným jádrům ARM. Základním postupem bylo se seznámit s použitím procesoru řady Kinetis M, na kterém byly aplikovány algoritmy na zpracování číslicového signálu. Cílem práce bylo popsat problematiku FIR a IIR filtrů, implementace těchto filtrů v programovacím jazyce C, jak s využitím matematického koprocesoru, tak bez něj včetně aplikací jednotlivých filtrů na měřený vstupní signál. Součástí práce je také zhodnocení dosažených výsledků.

Klíčová slova: FIR filtry, IIR filtry, Cortex M0+, MATLAB, matematický koprocesor, návrh filtru

Abstract

This work focuses on working with the core ARM Cortex M0 and its use and usability compared to other ARM cores. The basic procedure was conducted in order to become familiar with the use of the processor series Kinetis M, which was applied to the digital signal processing algorithms. The aim of the work was to describe the issues of FIR and IIR filters, implementing these filters in programming language C with and without using a mathematical coprocessor, including the applications of each filter to the measured input signal. Part of the work is also the evaluation of the results achieved.

Key Words: FIR filter, IIR filter, Cortex M0+, MATLAB, Mathematical Coprocessor, filter design

Obsah

Seznam použitých zkratek a symbolů

ADC	–	Analog to digital converter - analogově digitální převodník
A/D převodník	–	Analogové digitální převodník
CPU	–	Central processing unit - centrální procesorová jednotka
dB	–	decibel - logaritmická jednotka útlumu
D/A	–	digitálně analogový převodník
DP	–	Dolní propust
FIR	–	Finite impulse response - konečná impulsní odezva
HP	–	Horní propust
Hz	–	Hertz - jednotka frekvence
IIR	–	Infinite impulse response - nekonečná impulsní odezva
MCU	–	Mikrokontrolér
MMAU	–	Memory Mapped Arithmetic Unit - matematický koprocesor
ms	–	milisekundy - jednotka času
NDP	–	Normovaná dolní propust
PP	–	Pásmová propust
PZ	–	Pásmová zádrž

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

Úvod

Bakalářská práce se zabývá zpracováním FIR a IIR filtrů na daný vstupní signál pomocí jádra ARM Cortex M0+ s použitím matematického koprocesoru a bez něj. Cílem bylo se seznámit se základními funkcemi jádra ARM Cortex M0+, na kterém byly implementovány FIR a IIR filtry.

Úvod práce se věnuje vlastnostem a funkcím jádra ARM Cortex M0+ a procesoru řady Kinetis M. Následuje převod analogového signálu na digitální, kde jsou popsány různé typy A/D převodníků. Dále jsou popsány FIR a IIR filtry včetně jejich popisů metod návrhu. Návrhy těchto filtrů jsou prováděny v programu MATLAB. FIR a IIR filtry jsou implementovány v programovacím jazyku C v programu IAR Embedded Workbench s a bez použití matematického koprocesoru. V závěru byly vyhodnoceny získané výsledky a jednotlivé metody porovnány. Výsledkem je celkové zhodnocení trvání jednotlivých algoritmů.

1 ARM Cortex

Řada procesorů ARM Cortex-M je řada škálovatelná, kompatibilní a energeticky úsporná. Jsou to snadno použitelné procesory určené k pomoci vývojářům tak, aby vyhovovaly potřebám řídících aplikací. Tyto vlastnosti přináší více funkcí za nižší cenu, podporují konektivitu, lepší opětovné použití kódu a zlepšení energetické účinnosti.

Rodina Cortex-M obsahuje cenově a energeticky optimalizované MCU určené zejména pro práci se smíšenými signály, aplikace v Internet of Things, konektivitě, řízení motorů, inteligentní měření, zařízení s uživatelským rozhraním, automobilový průmysl, průmysl řídicích systémů, domácích spotřebičů, spotřebních výrobků a lékařských přístrojů.

Je to skupina 32-bitových RISC ARM procesorových jader s licencí vydanou společností ARM. V současné době tuto rodinu tvoří jádra Cortex-M0, M0 +, M1, M3, M4 a M7.

1.1 ARM Cortex M0+

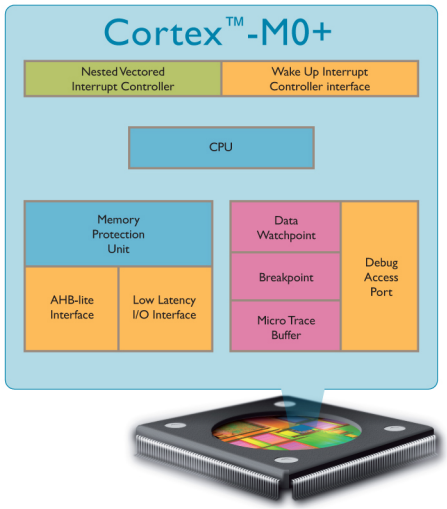
Procesor ARM Cortex-M0+ má velmi malý počet hradel, je to vysoce energeticky úsporný procesor, který je určen pro řídící mikrokontroléry a další vestavěné aplikace, které vyžadují cenovou a energetickou optimalizaci.

Vlastnosti a výhody procesoru

- Vysoká integrace systémových periferií zmenšuje plochu čipu a náklady na vývoj.
- Thumb-2 instrukční sada kombinuje vysokou hustotou kód s 32-bitovým výkonem.
- Podpora pro jednocyklový I/O přístup.
- Několik spánkových režimů.
- Integrované režimy spánku pro nízkou spotřebu energie.
- Rychlé vykonávání kódu umožňuje běh procesoru na nižší frekvenci nebo zvýšení doby režimu spánku.
- Optimalizované čtení kódu pro sníženou spotřebu FLASH a ROM.
- Obsahuje hardwarovou násobičku.
- Deterministická a vysoce výkonná práce s přerušením pro časově kritické aplikace.
- Deterministické časování instrukčních cyklů.
- Serial Wire ladění snižuje počet pinů potřebných pro ladění.

Funkčnost

Procesor Cortex-M0+ je konfigurovatelný, 32-bit RISC procesor s rozhraním AMBA AHB-Lite a obsahující vnitřní řadič NVIC. Má také volitelné hardwarové ladění, rozhraní pro jednocyklový přístup k I/O, a možnost zabezpečení paměti. Procesor vykonává Thumb-2 instrukce, který je kompatibilní s jinými procesory rodiny Cortex-M.



Obrázek 1.1: ARM Cortex M0+

1.2 ARM Cortex M3

Procesor ARM Cortex-M3 je výkonný 32-bitový procesor pro vysoce deterministické aplikace pracující v reálném čase. Procesor je vyvinut speciálně pro požadavky vysoce výkonných a zároveň levných platforem pro širokou škálu zařízení, jako jsou automobilové systémy, průmyslové řídící systémy a bezdrátové sítě a senzory.

Tento procesor nabízí vysoký výpočetní výkon a předvídatelnou odezvu systému na vnější a vnitřní události. Procesor je vysoce konfigurovatelný a umožňuje širokou škálu implementací od těch, kteří vyžadují zabezpečení paměti, levné zařízení vyžadující minimální prostor čipu. K dosažení tohoto cíle bylo potřeba provést v základní architektuře řadu zásadních změn. Zejména velmi zjednodušit proces tvorby softwaru. Procesor architektury Cortex-M3 lze tak použít
i v těch nejlevnějších aplikacích.

V procesorech Cortex-M3 je k dispozici nová sada příkazů s označením Thumb-2,
která programátorovi umožňuje dosažení až o 70 % vyššího výkonu na jeden megahertz než klasické ARM procesory, založené na jádře ARM7TDMI a vybavené pouze klasickou Thumb instrukční sadou. Navíc je možné dosáhnout až o 35% vyššího výkonu, než u shodných procesorů, které vykonávají základní sadu instrukcí ARM.

Tabulka 1: Srovnání funkcí procesorů ARM Cortex M0+ a ARM Cortex M3

Funkce	ARM Cortex M0+	ARM Cortex M3
ISA Support	Thumb® / Thumb-2 subset	Thumb® / Thumb-2
Pipeline	2 stupně	3 stupně
Výkon	2.46 CoreMarks/MHz	3.34 CoreMark/MHz
Výkon	0.95 / 1.11 / 1.36 DMIPS/MHz	1.25 / 1.50 / 1.89 DMIPS/MHz
Zabezpečení paměti	8 volitelných oblastí MPU	8 volitelných oblastí MPU
Přerušení	Nemaskovatelné přerušení (NMI) + 1 až 32 fyzických přerušení	Nemaskovatelné přerušení (NMI) + 1 až 240 fyzických přerušení
Režimy spánku	Integrované WFI a WFE instrukce a možnost Sleep On Exit (z přerušení) režimu. Dále Sleep & Deep Sleep možnosti spánku	Integrované WFI a WFE instrukce a možnost Sleep On Exit (z přerušení) režimu. Dále Sleep & Deep Sleep možnosti spánku.
Bitová manipulace	Bit Banding	Bit Banding
Rozšířené instrukce	Hardwarové jednocyklové (32x32) násobení	Hardwarové dělení (2-12 cyklů), Hardwarové jednocyklové (32x32) násobení, Podpora matematických operací se saturací..

V kapitole 2 bylo čerpáno z: [1],[2],[3],[4]

2 Procesory rodiny Kinetis M

Zařízení rodiny Kinetis M (KM) jsou 32-bit mikrokontroléry vyrobeny 90nm Thin Film Storage (TFS) technologií.

Tato zařízení jsou primárně zaměřena na trh měřících zařízení, jako jsou chytré jednofázové elektroměry pro Indii, Čínu a země Evropské unie a pro dvoufázové elektroměry pro USA a Japonsko.

KM zařízení jsou založena na 32-bitovém jádře ARM Cortex-M0+ s integrovaným Analog front-endem (AFE).

Frekvence CPU na těchto zařízeních může dosáhnout až 75MHz.

Tyto zařízení zahrnují vysoce přesný sigma delta A/D převodník, programovatelné zesilovače (PGA), matematický koprocesor, vysoce přesnou referenci vnitřního napětí, flash pamět, paměť RAM, fázový kompenzační logický blok a další periferie.

KM rodina zajišťuje tampering a obsahuje periferii pro přesné hodiny reálného času na všech zařízeních.

2.1 TWR-KM34Z50M

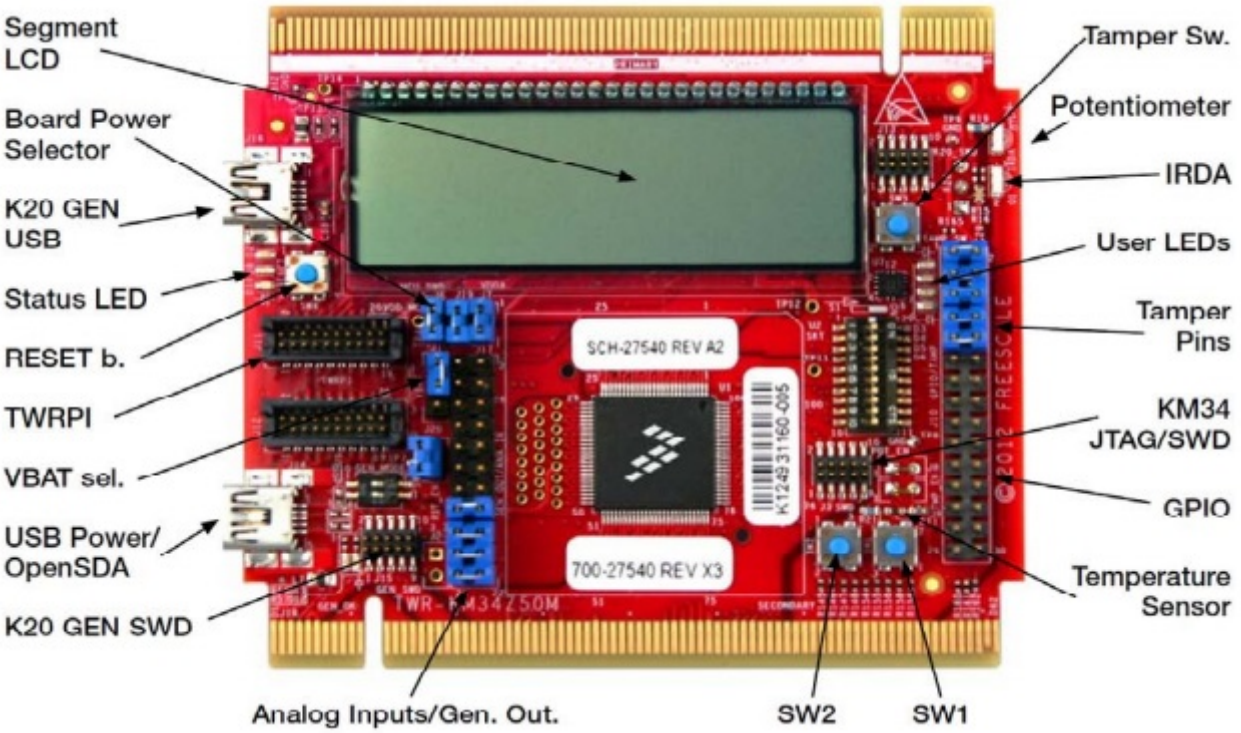
TWR–KM34Z75M MCU modul je navržený pro práci, a to buď v samostatném režimu nebo, jako součást systému Freescale Tower - modulová vývojová platforma, která umožňuje rychlý vývoj a nástroj opětovného použití prostřednictvím rekonfigurovatelného hardwaru.

TWR-KM34Z75M je vývojová platforma pro NXP KM3x a KM1x z rodiny mikrokontrolerů Kinetis.

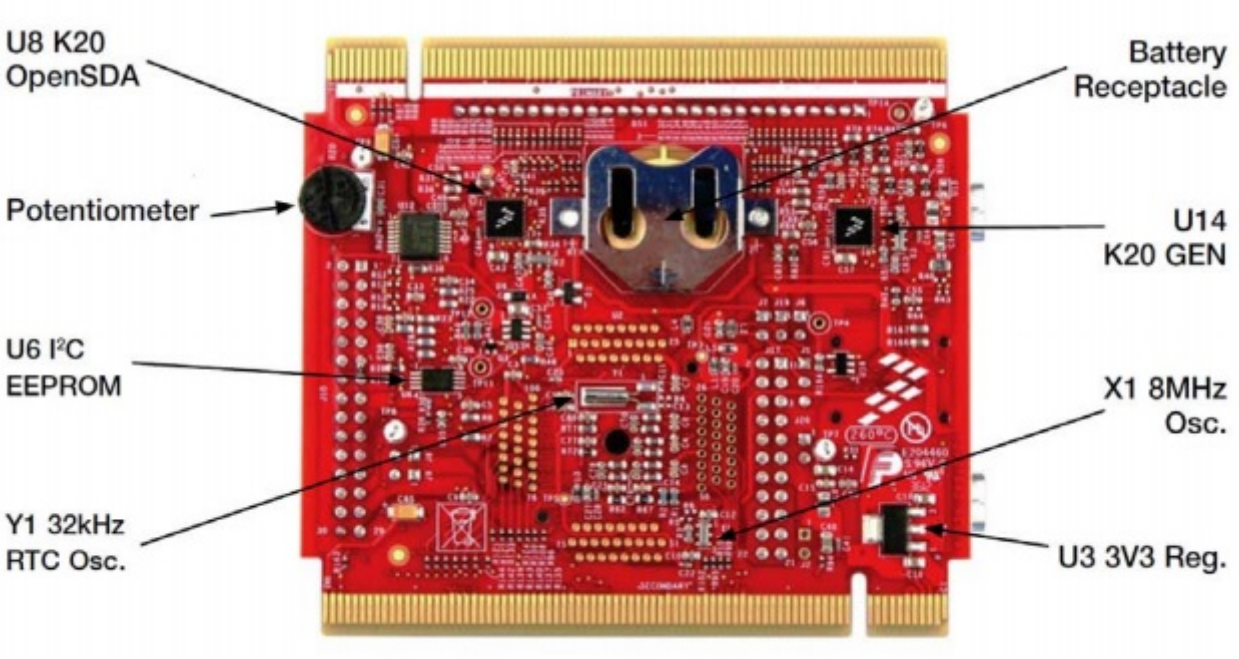
Funkce

- Procesor MKM34Z5256CLL5 MCU (75 MHz, 256 KB Flash, 16 KB RAM, nízká spotřeba, LQFP100 package)
- USB rozhraní s mini-AB USB konektorem
- Velké 160 - segmentové LCD
- Ladící obvod: open source JTAG/SWD (OpenSDA) s virtuálním sériovým portem
- Tříosý akcelometr / senzor proti neoprávněné manipulaci (MMA8491Q)
- Uživatelem řízené LED diody
- uživatelská tlačítka s GPIO přerušením
- 1 uživatelské tlačítko pro odhalení neoprávněné manipulace (tampering)
- 1 uživatelské tlačítko pro MCU reset

- Potenciometr
- Konektor pro GPIO a přístup k ADC
- Nezávislé napájení hodin reálného času (RTC)
- IRDA podpora
- NTC teplotní senzor
- Univerzální Tower Plug-in (TWRPI) socket



Obrázek 2.1: Kit - přední strana



Obrázek 2.2: Kit - zadní strana

V kapitole 3 bylo čerpáno ze zdroje: [5]

3 A/D převodníky

Analogově digitální převodník převádí analogový signál na digitální signál tak, aby umožnil zpracování na číslicových počítačích. V digitální podobě se signály kvalitněji zaznamenávají a přenášejí. D/A převodník zabezpečuje převod z digitálního signálu na analogový signál.

Při zpracování digitálního signálu musí být k dispozici posloupnost diskrétních vzorků x[n], se kterými se provádí diskrétní operace, jako např. analýza, filtrace, modulace. Spojité signály se zpracovávají digitálními technickými prostředky, které musí být převedeny na diskrétní posloupnost použitím vzorkování.

Vzorkování je proces, při kterém dochází k digitalizaci spojitého signálu tak, aby jeho určité hodnoty nebo úseky dostatečně reprezentovaly původní signál. Při tomto procesu mohou být ztraceny některé informace původního signálu. Pokud spojitý signál vzorkujeme dostatečně vysokou vzorkovací frekvencí f_{vz}, a kmitočtově ho omezíme, ke ztrátě informací nedochází nebo je přijatelně malá.

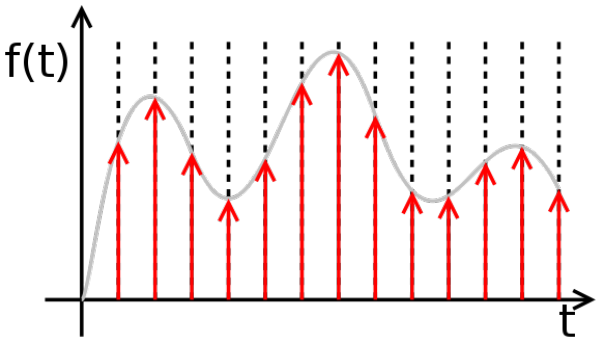
Protože počítače mají pouze konečnou kapacitu paměti, musíme se u reálného vzorkování omezit pouze na nezbytně nutné množství vzorků, které budeme dále zpracovávat.

Vzorkování se provede tak, že rozdělíme vodorovnou osu signálu na rovnoměrné úseky a z každého úseku odebereme jeden vzorek. Místo spojité čáry, kterou lze donekonečna zvětšovat, dostáváme množinu diskrétních bodů s intervalem odpovídajícím použité vzorkovací frekvence.

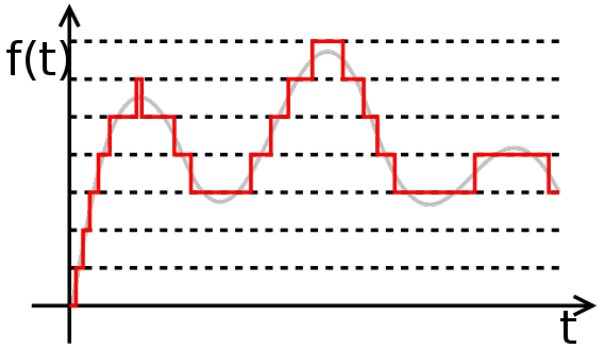
Po navzorkování signálu je získán konečný počet diskrétních vzorků původního analogového signálu. Vzorky jsou ale pro další zpracování nevhodné, protože stále obsahují mnoho informací. Dalším procesem jednorozměrné úrovně diskretizace je kvantování signálu.

Podstatou kvantování je zaokrouhlení hodnot signálu získaného při vzorkovaní na předem definované tzv. kvantizační úrovně. Hodnoty původního signálu se k dalšímu zpracování nepřenášejí přesně, ale vždy jako celočíselné hodnoty dané kvantizační úrovní, a tak se v převedeném digitálním signálu objeví kvantizační zkreslení. Zkreslení je ale ve srovnání s hodnotami daného původního signálu natolik malé, že jej divák nezaznamená.

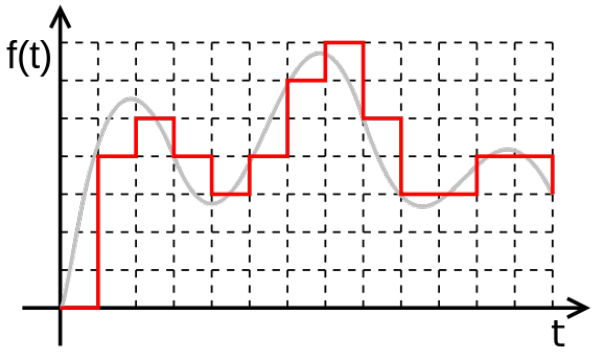
Na následujících obrázcích je zaznamenán proces převodu analogového signálu na digitální signál. Na obrázku 3.1 lze vidět proces vzorkování, na obrázku 3.2 proces kvantování a na obrázku 3.3 digitální signál.



Obrázek 3.1: Vzorkování



Obrázek 3.2: Kvantování



Obrázek 3.3: Digitální signál

K zajištění přesnosti musíme zajistit:

- Vzorkovací kmitočet musí být nejméně dvojnásobný než maximální kmitočet obsažený v analogovém signálu.
- Dostatečnou rychlost vzorkování - ta plyne z nejkratší možné doby převodu. Jde o důležitý parametr A/D převodníků.

- Pokud chceme mít malý kvantizační krok a malou kvantizační chybu, musí mít výstupní slovo dostatečný počet řádů.

3.1 Typy A/D převodníků

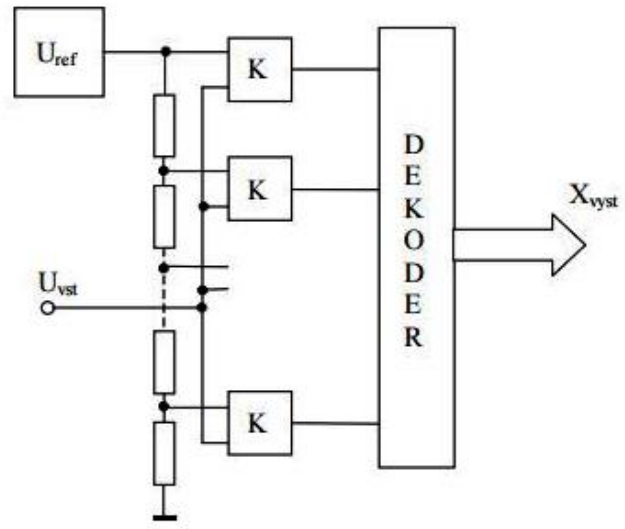
Mezi nejznámější typy A-D převodníku patří paralelní A/D převodník, převodník s postupnou aproximací, převodník s dvojitou integrací a Sigma-delta převodníky.

3.1.1 Paralelní A/D převodník

Převod probíhá v jednom časovém okamžiku, proto patří mezi nejrychlejší.

Referenční napětí se rozdělí na odporovém děliči složeném z 2+N odporů, kde N znázorňuje bitové rozlišení převodníku. Komparátory napětí srovnávají převáděné napětí s referenčním napětím. Pokud je Uvst $\geq$ Uref daného komparátoru, provede tento komparátor překlopení úrovně na výstupu. Převodník kódu hodnotí výstupy komparátorů a vykonává samotný převod na výstupní datové slovo.

Převodníky mají vysokou rychlost, jejich nevýhodou je poměrně velký počet komparátorů (pro n-bitový převodník je potřeba 2^{n-1} komparátorů). Tyto převodníky dnes působí v monolitické formě se všemi komparátory na čipu. Jejich rychlost se udává v jednotkách nanosekund.

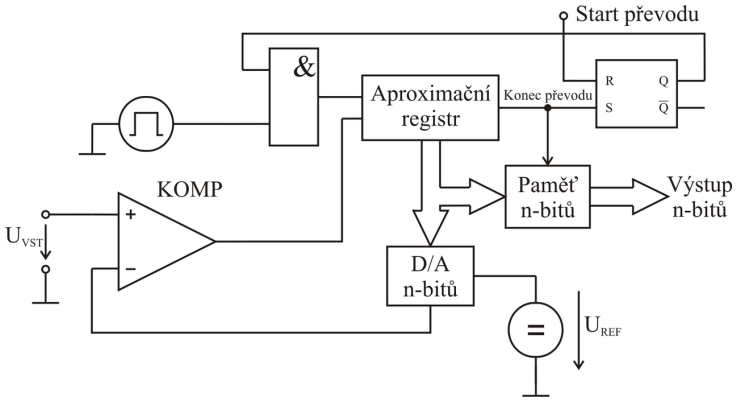


Obrázek 3.4: Paralelní A/D převodník

3.1.2 Převodník s postupnou aproximací

Převodník s postupnou aproximací obsahuje D/A převodník, komparátor, aproximační registr a výstupní registr. D/A převodník generuje pomocné porovnávací napětí, které komparátor porovnává se vstupním napětím.

Na začátku převodu aproximační registr seřídí v D/A převodníku hodnotu 1000 0000, která odpovídá polovině referenčního napětí u D/A převodníku $\frac{U_{ref}}{2}$. Pokud je $U_{vst} > \frac{U_{ref}}{2}$, v 1. bitu zůstane 1. Pokud ne, přepíše se na 0. V dalším kroku se nastaví na 1 další váhový bit. Na výstupu tedy bude 1100 0000 nebo 0100 0000 podle výsledku předchozího kroku. Znovu se porovnává $Uref/2$ s U_vst a aktuální bit se nastaví na 1 nebo 0. Tímto způsobem se postupně nastaví všechny bity.



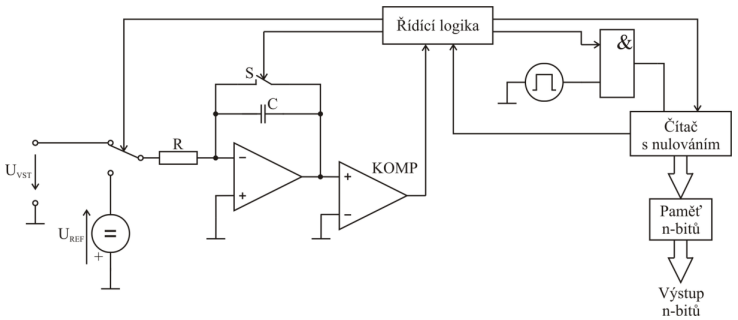
Obrázek 3.5: Převodník s postupnou aproximací

3.1.3 Převodník s dvojitou integrací

Převádí vstupní signál na jednobitový digitální signál. Používá se v multimetrech.

Na počátku měřícího cyklu je integrační kondenzátor vybitý a čítač vynulovaný. V prvním taktu je na vstup integrátoru přivedeno vstupní napětí a čítač čítá impulsy krystalového generátoru. Tím roste absolutní hodnota výstupního napětí integrátoru. Tento cyklus se opakuje, dokud nedojde k zaplnění čítače.

V okamžiku přetečení čítače přijde druhý takt. Přepínač se přepne a na vstup integrátoru se připojí referenční napětí, to má opačnou polaritu než vstupní napětí. Tím se snižuje absolutní hodnota výstupního napětí integrátoru. Jakmile dosáhne nulové hodnoty, překlopí komparátor, a tím se ukončí převod.



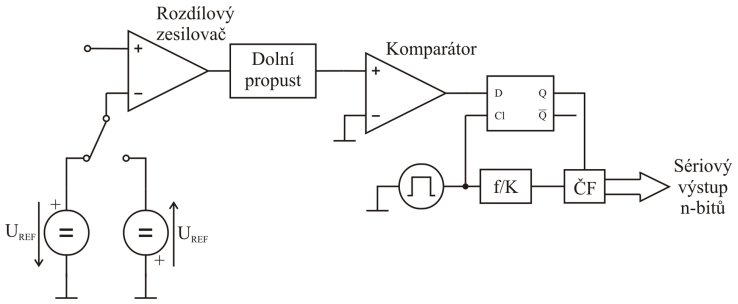
Obrázek 3.6: Převodník s postupnou aproximací

3.1.4 Sigma-delta A/D převodník

Sigma-delta ($\Sigma - \Delta$) převodníky jsou moderní převodníky pracující na principu $\Sigma - \Delta$ modulátoru, určeného pro převod analogového signálu na signál digitální. Vyznačuje se zejména jednoduchým hardwarem, ale složitějším softwarem pracující jako signálový procesor.

Převodník obsahuje sigma-delta modulátor a číslicový filtr. Základními obvody modulátoru jsou dolní propust (integrátor), napěťový komparátor a klopný obvod typu D, překlápěný hodinovým signálem s frekvencí f_0. Dále je zde přepínač dvouhodnotového signálu U_{ref}. Tento signál se odečítá od vstupního napětí v rozdílovém zesilovači.

Sigma-delta převodníky jsou vhodné pro měření stejnosměrných nebo pomalu se měnících napětí.



Obrázek 3.7: Sigma-delta AD převodník

V této kapitole bylo čerpáno ze zdroje: [7] a [6]

4 Filtry

Proces, ve kterém se mění složení vstupního signálu, se nazývá filtrace. Filtrace dokáže vymazat jednu nebo více složek frekvenční části ze spektra signálu, který je zpracováván. Filtr se může aplikovat na 1D a 2D signály.

Příkladem použití elektronických filtrů může být:

- Potlačení šumu – může být např. využit v radiových a zvukových signálech nebo také ve videosignálech,
- Zvýraznění frekvenčních pásem – příkladem toho je grafický ekvalizér, zvukové efekty, zaostření obrázků, a další,
- Omezení přenosového pásma v komunikačních kanálech – využívá se u ADSL a rozhlasového a televizního vysílání,
- Potlačení nebo odstranění specifických frekvencí – využívá se při blokování DC složky, aj.,
- Speciální operace jako jsou diferenciace, integrace, Hilbertova transformace a další.

4.1 Analogové filtry

Jsou aplikovány na spojité signály. Spojité signály jsou definovány v libovolném časovém okamžiku. Činnost spojitých soustav je popsána pomocí diferenciálních rovnic. Řešení diferenciální rovnice tvoří odezvu spojitého signálu na daný vstupní signál. Výpočet spektra je vyjádřen pomocí

Fourierovy transformace a přenos pomoci Laplaceovy transformace.

Vlastnosti

Jsou realizovány pomocí operačních zesilovačů, rezistorů a kondenzátorů. Dle teorie mají analogové filtry nekonečný frekvenční rozsah, ale praxe dokazuje, že můžou využívat maximální hodnotu v GHz.

Nevýhodou je citlivost na šum, nepřesnost, nestálost a nelinearita jejich součástek. Jsou omezené dynamickým rozsahem a mají špatnou výrobní reprodukovatelnost.

4.2 Číslicové filtry

Číslicové filtry představují jednu z možných realizací diskrétních systémů číslicovými technickými prostředky. Úkolem číslicových filtrů je stejně jako u analogových filtrů požadovaným způsobem ovlivnit kmitočtové spektrum vstupního signálu. Číslicový filtr zpracovává vstupní číslicový signál $x(n)$ tak, aby bylo dosaženo požadované výstupní posloupnosti $y(n)$.

Vnější popis číslicového filtru s konstantními koeficienty je dán lineární diferenční rovnicí obecně s-tého řádu s konstantními koeficienty viz 4.4. Tento popis u systému s jedním vstupem a jedním výstupem udává vztah mezi vstupním signálem *x(n)* a výstupním signálem *y(n)*. Vycházíme ze vztahu 4.5.

Vnitřní popis číslicového filtru definuje vztah mezi vstupními, výstupními a stavovými veličinami číslicového filtru. Stavové veličiny definují stav paměti mikroprocesoru, která číslicový filtr realizuje. Musíme znát vnitřní strukturu diskrétního systému nebo si musíme vytvořit její model. Stavový popis lze získat přímo z vnějšího popisu vhodnou úpravou přenosové funkce *H(z)*.

Číslicové filtry se implementují pomocí tří aritmetických operací, které jsou součet signálů (+), součin signálů (*) a zpoždění signálů (mov). Realizujeme je pomocí sčítačky, násobičky, a zpožďovacího členu, které jsou stavebními prvky pro realizaci číslicových filtru.

Číslicové filtry jsou vysoce lineární a mají flexibilní softwarovou implementaci, tj. můžou změnit parametry za běhu, také mají perfektní reprodukovatelnost a takřka neomezený dynamický rozsah (floating point). Mezi nevýhody patří, že digitální filtry vyžadují A/D a D/A převodníky a jejich frekvenční rozsah je dán vzorkovací frekvencí, která je dána Nyquistovým vztahem:

$$fr = fvz/2, \tag{4.1}$$

kde

fr = maximální frekvence

fvz = vzorkovací frekvence

Časový úsek číslicového signálu můžeme reprezentovat konečnou posloupností celých čísel. Obecně je n-tý prvek posloupnosti označován znakem *f(n)*, kde $n = ... - 2, -1, 0, 1, 2, ...$

Základním matematickým popisem vlastností diskrétních systémů jsou popsány diferenčními rovnicemi

$$b_s y(n+s) + b_{s-1} y(n+s-1) + b_{s-2} y(n+s-2) + \cdots + b_1 y(n+1) + b_0 y(n) = \tag{4.2}$$

$$= a_r y(n+r) + a_{r-1} y(n+r-1) + a_{r-2} y(n+r-2) + \cdots + a_1 y(n+1) + a_0 y(n),$$

kde

$b0, b1, \cdots, bs$ a $a0, a1, \cdots, ar$ jsou koeficienty diferenční rovnice

$x(n)$ je diskrétní vstupní signál

$y(n)$ je diskrétní výstupní signál

Má-li systém několik vstupů a výstupů, je nutné sestavit více diferenčních rovnic, pro každý vstup a výstup.

Výpočet přenosu je vyjádřen pomoci Z - transformace. Tu definujeme jako transformaci posloupnosti *f(n)*

$$F(z) = \sum_{n=0}^{+\infty} f(n) z^{-n}, \tag{4.3}$$

kde z je komplexní proměnná.

Posloupnost je schopná transformace Z, pokud řada 4.3 konverguje alespoň pro jedno komplexní z.

Použití Z-transformace:

- Z-transformací diferenční rovnice získáme algebraickou rovnici, ze které snadno odvodíme přenosovou funkci systému *H(z)*.
- Z-transformací impulsní charakteristiky *h(n)* získáme také přenosovou funkci systému *H(z)*.

Diskrétní Fourierova transformace je zvláštním případem Z-transformace s omezením na konečné posloupnosti. DFT byla zavedena pro periodické signály.

$$F(k) = \sum_{n=0}^{N-1} f(n) e^{-jnk\frac{2\pi}{N}}, \tag{4.4}$$

kde k = 0,1, $\cdots$, N-1.

Fourierova transformace neslouží jen k popisu signálů, ale i k výpočtu Fourierova integrálu a hlavně k výpočtu spekter signálů.

Přenosová funkce číslicového filtru je racionální lomená funkce definovaná v Z-rovině, která se vyjádří vztahem:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{a_0 + a_1 z + a_2 z^2 + \cdots + a_r z^r}{b_0 + b_1 z + b_2 z^2 + \cdots + b_s z^s}, \tag{4.5}$$

Stručně ji pak můžeme zapsat ve tvaru

$$Y(z) = H(z)X(z). \tag{4.6}$$

Podle přenosové funkce dělíme filtry na FIR a IIR filtry.

V této kapitole jsem čerpala z těchto zdrojů[9], [10], [11]

4.2.1 FIR filtry

FIR znamená Finite Impulze Response, filtry s tzv. konečnou impulsovou odezvou, tj. má konečný počet nenulových výstupních hodnot po vybuzení filtru jednotkovým impulsem.

Jde o číslicové filtry, které filtrují již plně digitální signál v podobě posloupnosti vzorků po převodu analogového signálu A/D převodníkem. K jejich praktické realizaci je potřeba procesor.

Mezi jeho výhody patří jednoduchá struktura, jednoduchý návrh a testování již realizovaných filtrů. Koeficienty filtru představují impulsovou odezvu. FIR filtr je nerekurzivní, a tudíž je vždy stabilní. Nemůže nikdy dojít k rozkmitání.

Jeho nevýhodou je obvykle velký řád filtru, tj. velký počet koeficientů, a tím i rozsáhlá struktura filtru. Pro velké řády filtru je i vysoká výpočetní časová náročnost. V nepropustném pásmu má omezené nastavení útlumu.

U FIR filtrů lze dosáhnout lineární fáze v celém kmitočtovém pásmu:

- Symetrickou impulzní odezvou

$$h(n) = h(N - 1 - n), \tag{4.7}$$

- Antisymetrickou impulzní odezvou

$$h(n) = -h(N - 1 - n), \tag{4.8}$$

kde

N je počet reálných koeficientů

N-1 je řád filtru

$n \in < 0, N - 1 >$

Diferenční rovnice FIR filtru je ve tvaru:

$$y(n) = \sum_{k=0}^{N-1} h_k x_{n-k}. \tag{4.9}$$

Diferenční rovnice, která popisuje FIR filtr, je konvoluční suma. Výstupní signál je dán konvolucí mezi koeficienty impulzní odezvy h a vzorky vstupního signálu x. Proto se FIR filtry označují jako konvoluční filtry.

Přenosová funkce je daná Z-transformací impulzní odezvy filtru:

$$H(z) = \sum_{k=0}^{N-1} h(k) z^{-k}. \tag{4.10}$$

Všechny póly v Z-rovině leží v počátku ($z_\infty(k) = 0$), pro všechny k.

4.2.1.1 Metody návrhu FIR filtrů

Problémy při návrhu FIR filtru:

- Specifikace filtru – před návrhem filtru upřesníme nezbytné vlastnosti filtru ve frekvenční oblasti.
- Aproximace specifikovaného chování filtru matematickými metodami – aproximace filtru je nejdůležitější část návrhu filtru.
- Implementace navrženého filtru – tento krok je výsledkem specifikace a aproximace navrženého číslicového filtru, který je ve tvaru diferenční rovnice, přenosové funkce *H(z)* nebo impulzní odezvy *H(n)*. Implementace filtru probíhá pomocí technologických prostředků nebo programem.

Nejčastěji se používají tyto metody návrhu:

- Metoda váhové posloupnosti
- Metoda vzorkování kmitočtové charakteristiky
- Optimalizační metody se stejnosměrným zvlněním

Metoda váhové posloupnosti Při návrhu vycházíme ze specifikace ideální frekvenční charakteristiky $H_d(e^{j\omega})$. Kmitočtová charakteristika číslicového filtru je periodickou funkcí úhlového kmitočtu s periodou 2π.

Aby požadovaný filtr byl realizovatelný, musíme ideální impulzní odezvu $h_d(n)$ omezit z obou stran na konečný počet vzorků N a impulsní odezvu $h_d(n)$ vynásobíme vhodnou funkcí okna *w(n)*, která má délku *N*.

$$h_w(n) = hd(n) \cdot w(n). \tag{4.11}$$

Následkem omezení dojde k chybám ve skutečné frekvenční charakteristice oproti ideální charakteristice. Pro správnou realizaci, která vede k realizovatelnému filtru, je zaveden odpovídající posuv impulsní charakteristiky o $M = (N-1)/2$ vzorků.

Impulsní charakteristika bude mít tvar:

$$H(n) = \{\frac{sin[\omega_p(n-m)]}{\pi(n-M)}, \tag{4.12}$$

pro $0 \leq n \leq N-1$ a $M = \frac{N-1}{2}$

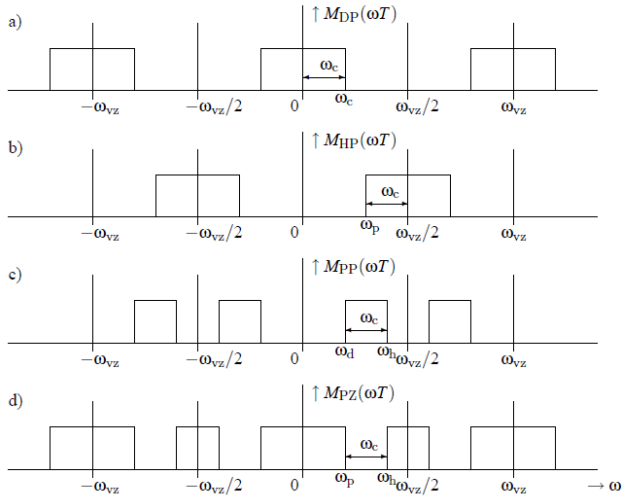
$$H(n) = 0, \tag{4.13}$$

pro všechny ostatní n.

Ideální příklady modulů kmitočtové charakteristiky pro číslicové filtry jsou dolní propust, horní propust, pásmová propust a pásmová zádrž viz obrázek níže.

Přenosová funkce je definovaná v intervalu $(-\pi, \pi)$ a je periodická s periodou 2π.

Na následujícím obrázku jsou zaznamenány ideální příklady modulů dolní propusti, horní propusti, pásmová propusti a pásmové zádrže.



Obrázek 4.1: Ideální příklady modulů a)Dolní propust b)Horní propust c)Pásmová propust d)Pásmová zádrž

Pro danou specifikaci zvolíme nebo vypočteme délku N a volíme typ okna $w(n)$ s nejširším hlavním lalokem, který odpovídá specifikaci.

Chceme-li minimalizovat chybu pomocí Čebyševovy aproximace, definujme požadovanou amplitudovou charakteristiku $H_{dr}(\omega)$ a váhovací kmitočtovou funkcí $W(\omega)$. Váhovací funkce umožní nezávislé ovládání zvlnění δ_1, δ_2.

Typy oken	Šířka pásma $\Delta\omega$	Šířka hlavního laloku	Vrchol postranního laloku [dB]	Minimální odstup [dB]
Obdélníkové	$\frac{1{,}8\cdot\pi}{N}$	$\frac{4\cdot\pi}{N\cdot T}$	-13	21
Trojúhelníkové	$\frac{6{,}1\cdot\pi}{N}$	$\frac{8\cdot\pi}{N\cdot T}$	-25	25
Hannovo	$\frac{6{,}2\cdot\pi}{N}$	$\frac{8\cdot\pi}{N\cdot T}$	-31	44
Hammingovo	$\frac{6{,}6\cdot\pi}{N}$	$\frac{8\cdot\pi}{N\cdot T}$	-41	53
Blackmanovo	$\frac{11\cdot\pi}{N}$	$\frac{12\cdot\pi}{N\cdot T}$	-57	74

Tabulka 2: Vlastnosti váhových oken

Metoda vzorkování kmitočtové charakteristiky

Tato metoda se využívá zvláště pro úzkopásmové filtry, ale její význam se postupně ztrácí vlivem Remezovu algoritmu.

Vyjdeme z ideální kmitočtové charakteristiky filtru $H(e^j\omega)$, na který vzorkujeme námi vybraný řád N, který je v ekvidistantních kmitočtech přes celé kmitočtové pásmo $(0, 2\pi)$ a získáme tak diskrétní funkci *H(k)*.

Impulsní charakteristika je dána ze vztahu

$$h(n) = \frac{1}{N} \sum_{k=0}^{N-1} H(k) e^{\frac{2\pi nk}{N}}. \tag{4.14}$$

Metoda spektrálního vzorkování je jednoduchá, rychlá, ale poskytuje malou kontrolu nad průběhem výsledné frekvenční charakteristiky. Pro řízený návrh je nutné vzorkovat i v přechodovém pásmu. V bodech frekvenčních vzorků výsledná odezva souhlasí s původní ideální, v ostatních je dána spojitou funkcí $|H(j\omega)|$.

Optimalizační metody se stejnosměrným zvlněním

Optimalizační metody poskytují z uvedených klasických metod nejlepší dosažitelné výsledky pro daný stupeň přenosové funkce číslicového filtru typu FIR s lineární fázovou charakteristikou. U většiny ostatních metod je aproximační chyba největší na okraji přechodových pásem a menší v oblastech vzdálených od přechodového pásma. Intuitivně se zdá být vhodnější rozprostřít rovnoměrně aproximační chybu přes celé propustné a nepropustné pásmo.

Tato metoda je často používána, protože dává dobré výsledky, a to nejen pro klasické typy filtrů, jako jsou filtry typu DP, HP, PP nebo PZ, ale i pro diferenciátory (modul v propustném pásmu lineárně narůstá) a Hilbertovy transformátory (vytváření komplexního analytického signálu), vše z lineární fází.

Pro filtry FIR s lineární fázi lze odvodit soubor podmínek, pro které je návrh filtru optimální ve smyslu minimalizace maximální chyby aproximace. Často je tento postup nazýván minimax metoda nebo Čebeševova aproximace.

Filtry FIR, které mají tuto vlastnost, se nazývají se stejnosměrným zvlněním, protože chyba aproximace je u nich rozložena rovnoměrně v propustném i nepropustném pásmu.

Problém minimax

Použitím trigonometrických vztahů popsali Parks a McClellan každý výraz pro amplitudovou charakteristiku $H_r(\omega)$ jako součin funkce $R(\omega)$ a $P(\omega)$.

$$H_r(\omega) = R(\omega) \cdot P(\omega). \tag{4.15}$$

Aproximační funkce $P(\omega)$ má obecný tvar

$$P(\omega) = \sum_{m=0}^{L} \alpha(m) cos\omega m, \tag{4.16}$$

kde
L značí řád aproximačního polynomu
$P(\omega)$ a $\alpha(m)$ značí parametry filtru.

Chceme-li minimalizovat chybu pomocí Čebyševovy aproximace, definujme požadovanou amplitudovou charakteristiku $H_dr(\omega)$ a váhovací kmitočtovou funkcí $W(\omega)$. Váhovací funkce umožní nezávislé ovládání zvlnění δ_1, δ_2.

Důležitý parametr optimalizace je počet extrému maxim a minim, který bude vykazovat chybová funce $E(\omega)$. Parks a McClellan aplikovali Čebeševovu polynomiální aproximaci a shrnuli ji do tzv. Alternačního teorému.

Alternační teorém

Je-li $P(\omega)$ trigonometrický polynom L-tého řádu, potom nutnou a postačující podmínkou proto, aby $P(\omega)$ byl jednoznačnou aproximací funkce $H_dr(\omega)$ na uzavřeném intervalu $< 0, \pi >$. Je požadavek, aby chybová funkce $E(\omega)$ vykazovala nejméně $(L + 2)$ zákmitů (alternací) na $(L + 2)$ kmitočtech ω_i na S takových, že

$$E(\omega_i) = E(\omega_{i-1}), \text{pro } \omega_0 < \omega_1 < \cdots < \omega_{L+1} \in S \tag{4.17}$$

Většina optimalizovaných, stejnoměrně zvlněných filtrů má *L+2* extrémů zvlnění. Pro některé kombinace kmitočtů ω_p a ω_p může vykazovat i *L+3* extrémů.

Parks McClellanův algoritmus

Z alternačního teorému vyplývá, že řešení aproximace minimax existuje. Neříká však nic o tom, jak toto řešení získat. Délka FIR filtru N (resp. L) totiž není známá, ani kmitočty extrémů ω_i, ani parametry $\alpha(n)$, ani maximální chybu δ.

Parks a McClellan provedli iterativní řešení použitím Remezova algoritmu. Ten předpokládá, že známe délku filtru N (nebo L) a poměr $\frac{\delta_2}{\delta_1}$. Při specifikaci filtru je obvykle zadané $\delta, \delta_1, \delta_2, \omega_p, \omega_s$.

V kapitole 5 byly použity tyto zdroje [9],[10],[11]

4.2.2 IIR filtry

IIR znamená Infinite Impulse Response, filtry s tzv. nekonečnou impulzní odezvou, tj. má nekonečnou impulzní odezvu a vyžaduje minimálně jednu zpětnovazební vazbu.

Tyto filtry mohou být nestabilní, proto musí být stabilita vždy ověřena. Filtr bude stabilní, pokud všechny póly leží uvnitř jednotkové kružnice. ($|z_\infty[k]| < 1$, pro všechny k). Nulové body lze umisťovat uvnitř i vně jednotkové kružnice, jsou souměrné podle jednotkové kružnice a ovlivňují chování filtru v propustném pásmu. Lze dosáhnout velmi strmé přechody mezi propustným a nepropustným pásmem, a to i při malém řádu filtru.

IIR filtry mají ve srovnání s FIR filtry nižší výpočetní nároky. Mezi jeho nevýhody patří, že nemůžou dosáhnout lineární fázové kmitočtové charakteristiky v celém rozsahu a jsou velmi citlivé na kvantování.

Diferenční rekurzivní rovnice mezi vstupem x_n a výstupem y_n je ve tvaru:

$$y(n) = \sum_{i=0}^{r} L_i x_(n-i) - \sum_{i=1}^{m} K_i y_{n-i}, \tag{4.18}$$

kde

L_i, K_i jsou systémové koeficienty v dopředných či zpětných vazbách,
r udává počet zpoždění v nerekurzivní složce rovnice,
m je řád systému a rovněž počet zpoždění pro rekurzivní složku.

Přenosová funkce je ve tvaru racionální lomené funkce

$$H(z) = \frac{\sum_{i=0}^{r} L_i z^(m-1)}{\sum_{i=0}^{m} K_i z^{m-1}} = A\frac{b_0 \prod_{i=1}^{r}(z-n_i)}{\prod_{i=1}^{m}(z-p_i)}, \tag{4.19}$$

kde

$K_0 = 1$
p_i jsou nulové body, které realizují rekurzivní část
n_i jsou nulové body, které realizují nerekurzivní část
A je zesílení.

4.2.2.1 Metody návrhu číslicových filtrů typu IIR

Metody návrhu číslicových filtrů typu IIR lze rozdělit na dvě skupiny:

1. **Metody využívající analogových prototypů** – tj. souvislost mezi rovinou p (laplaceova transformace) a rovinou z (transformace Z). Mezi tyto metody patří bilineární transformace, metody signálové invariance apod.

2. **Metody přímého návrhu v rovině z bez návaznosti na analogové ekvivalenty** - mezi tyto metody patří metoda nejmenších čtverců v časové a kmitočtové oblasti, identifikační parametrické metody používající modelů typu ARMA (Auto Regressive Moving Average) a další.

4.2.3 Metody využívající analogových prototypů

Nejčastěji se používají metody, které využívají analogových prototypů. Tyto metody využívají aproximační úlohy pro aproximaci modulu normované ideální propusti (NDP), která se poté transformuje na požadovaný typ filtru.

Číslicový filtr je dán převodem analogového prototypu přenosové funkce $H(p)$ z p-roviny do z-roviny diskrétního systému. Touto operací získáme výslednou přenosovou funkci $H(z)$.

Vlastnosti funkcí aproximujících ideální požadovanou kmitočtovou charakteristiku NDP:

- **Butterworthova aproximace:**

  Je charakteristická tím, že má maximálně plochou amplitudovou charakteristiku, a proto je její překmit nulový. Strmost přechodového pásma je ze všech aproximací nejmenší a fázová charakteristika se blíží nejvíce k lineárnímu průběhu. Jeho analogový prototyp má pouze nulové póly.

- **Čebyševova aproximace 1. typu:**

  U této aproximace dochází ke zvlnění modulové kmitočtové charakteristiky v propustném pásmu, v nepropustném pásmu je charakteristika hladká. Analogový prototyp má jako u Butterworthovy aproximace pouze nulové póly.

- **Čebyševova aproximace 2. typu:**

  U Čebyševovy aproximace 2. typu na rozdíl od 1.typu dochází naopak ke zvlnění v nepropustném pásmu, zatímco v propustném pásmu je charakteristika hladká. Analogový prototyp má nulové body i póly.

- **Cauerova aproximace**

  U této aproximace dochází ke zvlnění modulu kmitočtové charakteristiky v nepropustném i propustném pásmu a charakteristika vykazuje proměnný počet lokálních extrémů podle řádu systému. Analogový prototyp má jak nulové body, tak i póly. Fázová kmitočtová charakteristika se u této aproximace nejvíce odlišuje od lineárního průběhu.

4.2.3.1 Invariantní impulsní odezva:

Pod pojmem invariantní impulsní odezva rozumíme, že k impulsní odezvě $h_a(t)$ známého analogového filtru hledáme impulsní odezvu číslicového filtru $h(n)$, která se shoduje ve vzorkovací periodě $t = nT$ s analogovou předlohou.

$$h_d(n) = h_a(nT), \tag{4.20}$$

kde

T je perioda vzorkování.

Rozklad analogové přenosové funkce na parciální zlomky

$$h_a(p) = \frac{Q(p)}{N(p)} = \sum_{\mu=1}^{n} \frac{k_\mu}{p - p_u}, \tag{4.21}$$

s příslušnou impulsní odezvou

$$h_d(n) = \sum_{\mu=1}^{n} k_\mu e^{p\mu t}, \tag{4.22}$$

je nahrazen přenosovou funkcí číslicového filtru, jenž má shodnou impulsní odezvu v bodech $t = nT$ a tedy

$$H_d(n) = \sum_{\mu=1}^{n} \frac{k_\mu}{1 - e^{p\mu T} z^{-1}} \tag{4.23}$$

Kmitočtová charakteristika číslicového filtru je rovna kmitočtové odezvě periodicky vzorkované analogové přenosové funkce a tedy

$$H_d(e^j\omega) = \frac{1}{T} \sum_{n=-\infty}^{\infty} H_a(\frac{j\omega}{T+j} + \frac{2\pi}{Tn}) \tag{4.24}$$

Ze vzorkovacího teorému je zřejmé, že jen tehdy, je-li analogová přenosová funkce omezená $H_a(j\omega) = 0$ pro $|\omega| \geq \frac{\pi}{T}$ jsou kmitočtové odezvy přenosových funkcí shodné

$$H_a(e^{j\omega}) = 1/T H_a(j\frac{\omega}{T}) \tag{4.25}$$

V praktických situacích tomu tak nebývá a dochází k překrývání spekter transformovaných přenosových funkcí.

Tato metoda je vhodná jen pro transformace analogových filtrů se shora omezenou kmitočtovou charakteristikou.

4.2.3.2 Bilineární transformace:

Provede se substituce argumentu s přenosové funkce $H(s)$ analogového prototypu za argument z podle vztahu:

$$s = \frac{2}{T}\left(\frac{1 - z{-}1}{1 + z{-}1}\right) \tag{4.26}$$

Transformace kmitočtů z analogové do číslicové oblasti je nelineární. Nekonečná analogová frekvenční osa se transformuje na konečný úsek číslicové frekvenční osy. Tzn., že během návrhu musíme provést předkreslení frekvenční osy pro všechny frekvence:

$$\omega = \frac{2}{T} arctg \frac{\omega_a T}{2}, \omega_a = \frac{2}{T} tg \frac{\omega T}{2} \tag{4.27}$$

Metoda bilineární transformace je vzhledem ke zkreslení kmitočtové osy vhodná pro návrh filtrů s po částech konstantní modulovou kmitočtovou charakteristikou (tj. filtry typu DP, HP, PP nebo PZ). V některých případech lze nelinearity využít k dosažení větší strmosti v přechodovém pásmu číslicového filtru. Pro návrh filtru s lineárně rostoucí nebo klesající hodnotou modulu kmitočtové charakteristiky není použití této metody příliš vhodné.

4.2.4 Metody přímého návrhu v rovině *z* bez návaznosti na analogové ekvivalenty

Metody přímého návrhu se vzhledem ke své výpočtové náročnosti používají ve vybraných případech, ve kterých je nutné dosáhnout lepších výsledků než při použití metod vycházejících z analogových prototypů.

4.2.4.1 Metoda nejmenších čtverců:

U této metody nalezení koeficientů filtrů s požadovanou frekvenční charakteristikou převedeme na řešení lineárních a nelineárních rovnic. U této metody je řešení soustav složité a výpočetně náročné.

4.2.4.2 Kmitočtová transformace:

Tato metoda umožňuje transformaci digitálního filtru typu dolní propust na libovolný jiný typ číslicového filtru (DP, HP, PP, PZ). Jedná se o nalezení takového zobrazení q(z), která nemění průběh modulové charakteristiky v propustném a nepropustném pásmu, ale mění pouze polohu okrajů propustného pásma.

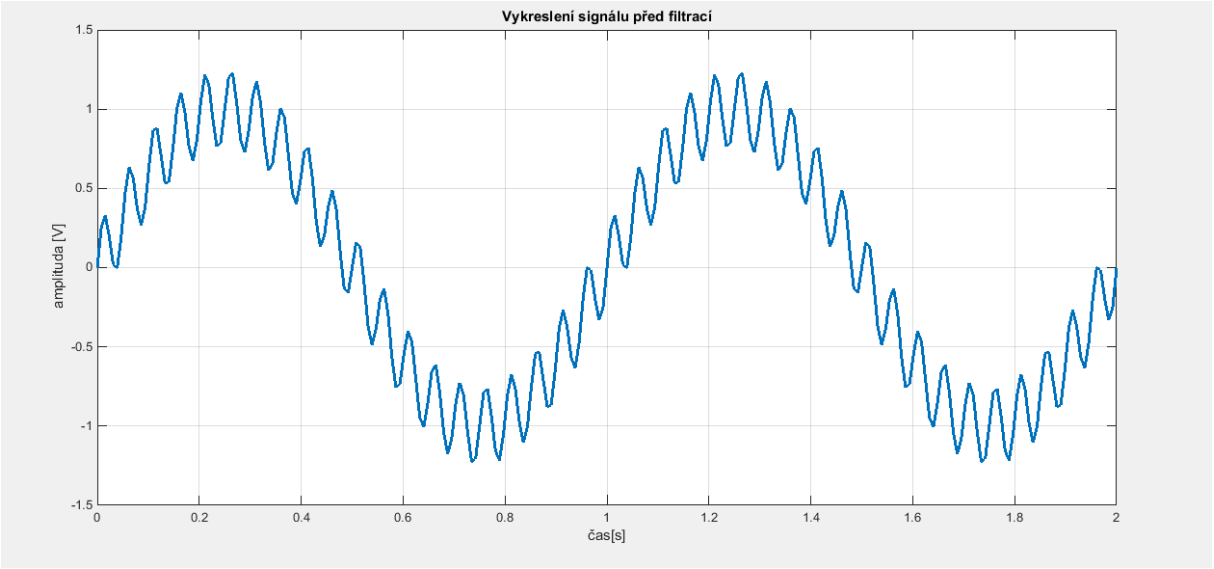
V této kapitole vylo čerpáno z [8], [9], [10].

5 Návrh FIR a IIR filtrů v programu MATLAB

Matlab je program, který poskytuje řadu matematických metod pro analýzu dat, vykreslování 2D nebo 3D grafů funkcí, implementaci algoritmů počítačovou simulaci i vytváření aplikací včetně uživatelského rozhraní. Díky svým funkcím byl použit pro návrh filtru.

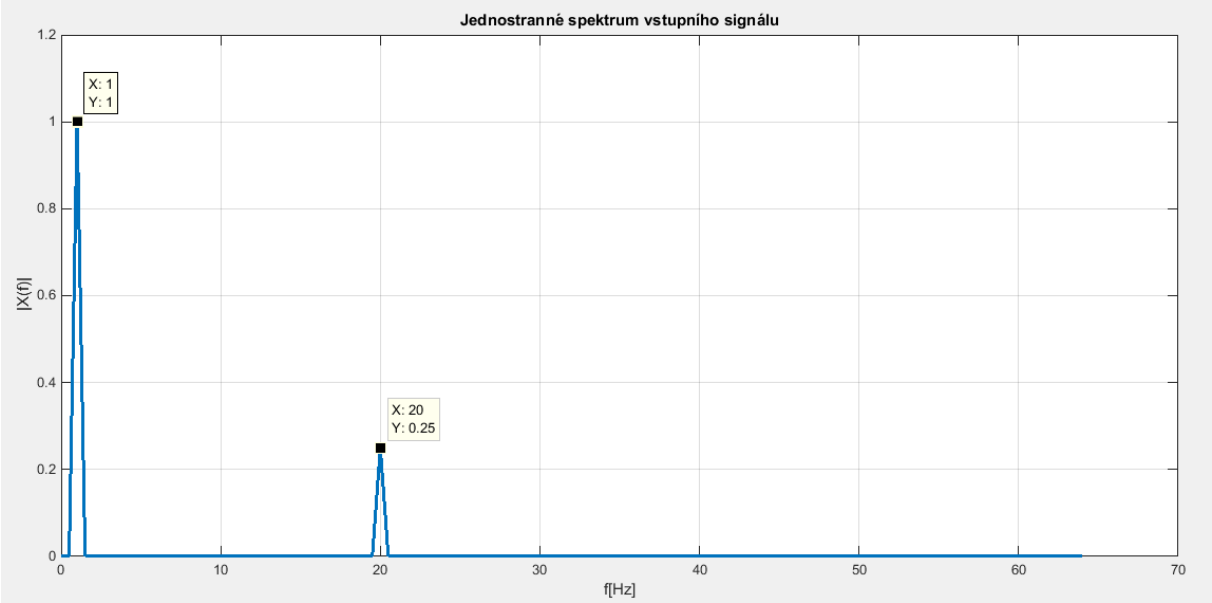
5.1 Návrh IIR filtru

Při návrhu IIR filtru byl zvolen filtr typu dolní propust a signál $1 \cdot sin(2 \cdot \pi \cdot t) + 0,25 \cdot sin(2 \cdot \pi \cdot 20 \cdot t)$, kde druhá sinusovka představuje šum (druhá sinusová křivka bude dále nazývána šum), který má být ze signálu odstraněn. Vzorkovací frekvence byla zvolena 128Hz, a cutoff frekvence 5Hz. Na následujícím obrázku můžeme vidět původní vstupní signál.



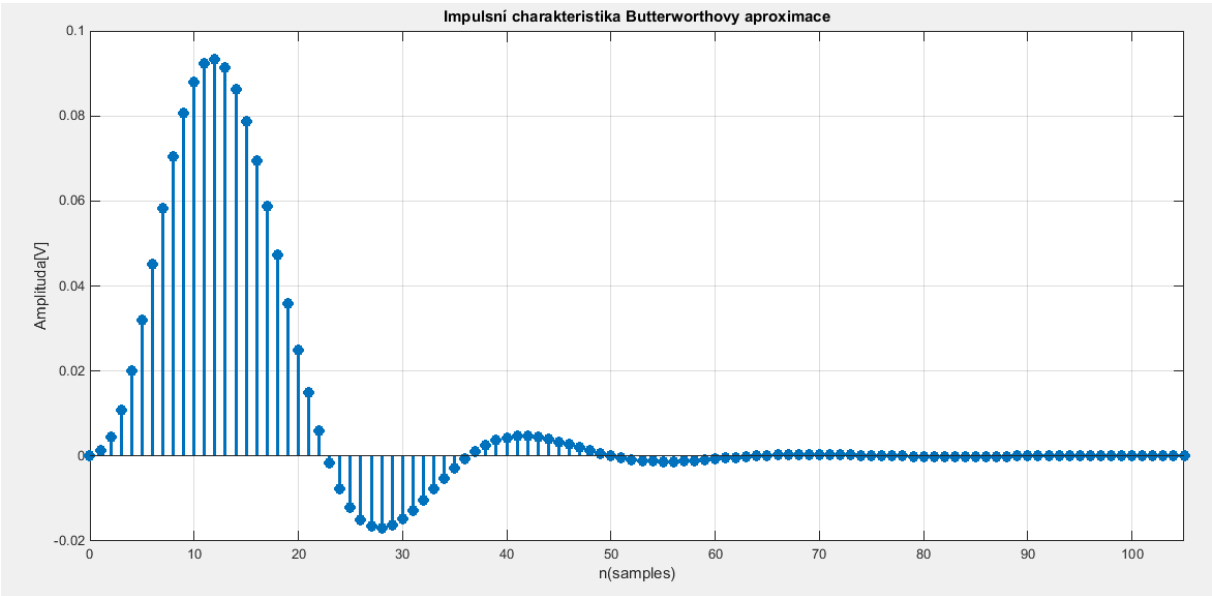
Obrázek 5.1: Vstupní signál

Na obrázku 5.2 je zobrazeno jednostranné amplitudové spektrum vstupního signálu.



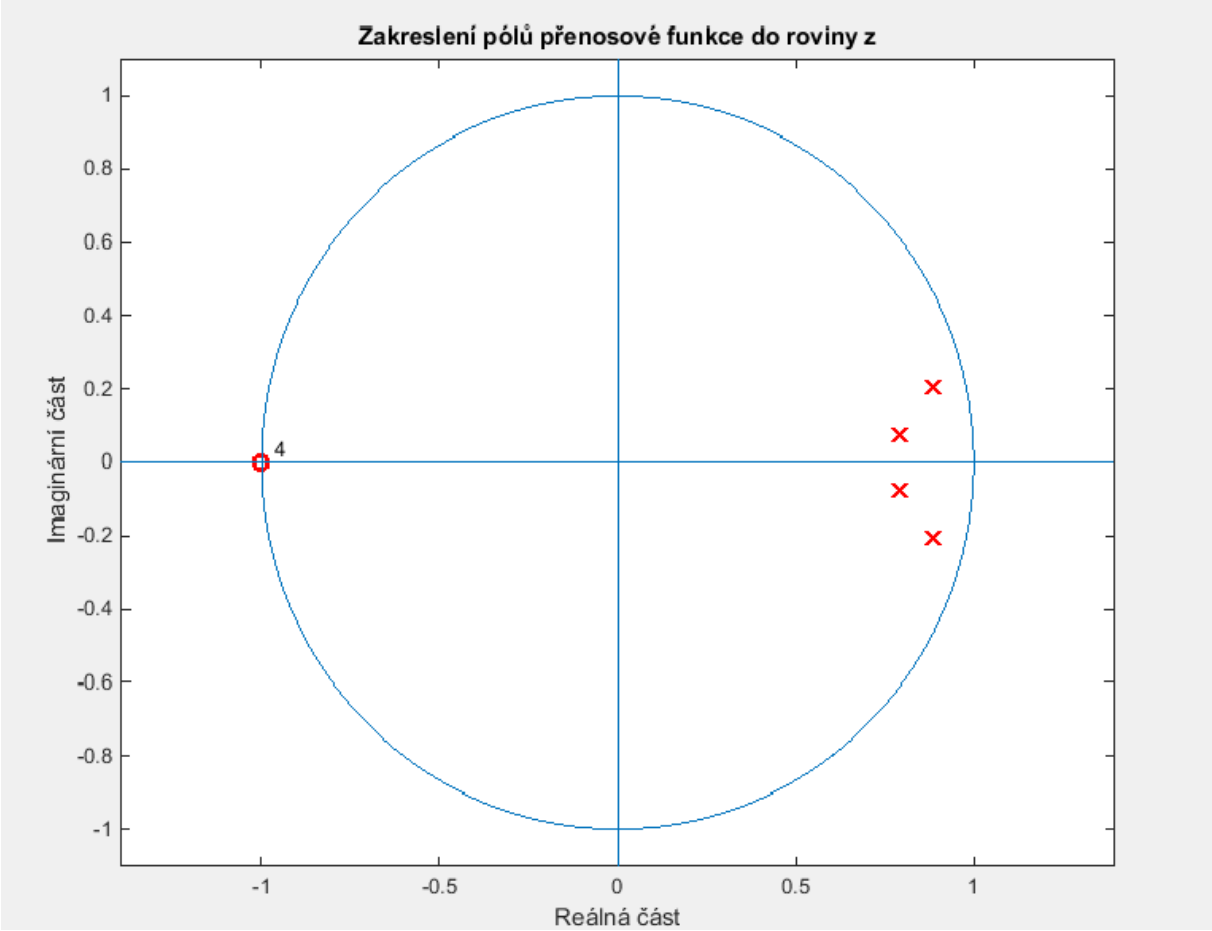
Obrázek 5.2: Jednostranné amplitudové spektrum vstupního signálu

Jako analogový prototyp byla zvolena Butterworthova aproximace ideální dolní propusti. Řád filtru byl zvolen N=4. Na obrázku můžeme vidět impulsní charakteristiku Butterworthovy aproximace.



Obrázek 5.3: Impulsní charakteristika Butterworthovy aproximace

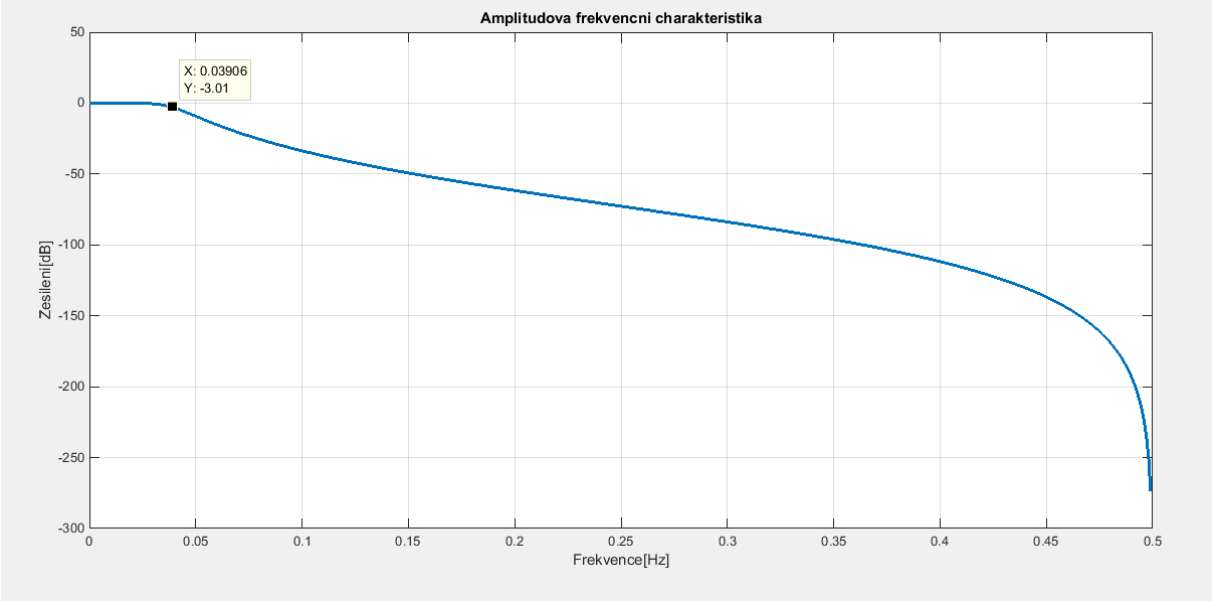
Jak už bylo zmíněno v kapitole 4.2.2, při návrhu IIR filtru je třeba kontrolovat stabilitu filtru pomocí zakreslení pólů přenosové funkce do roviny z.



Obrázek 5.4: Stabilita IIR filtru

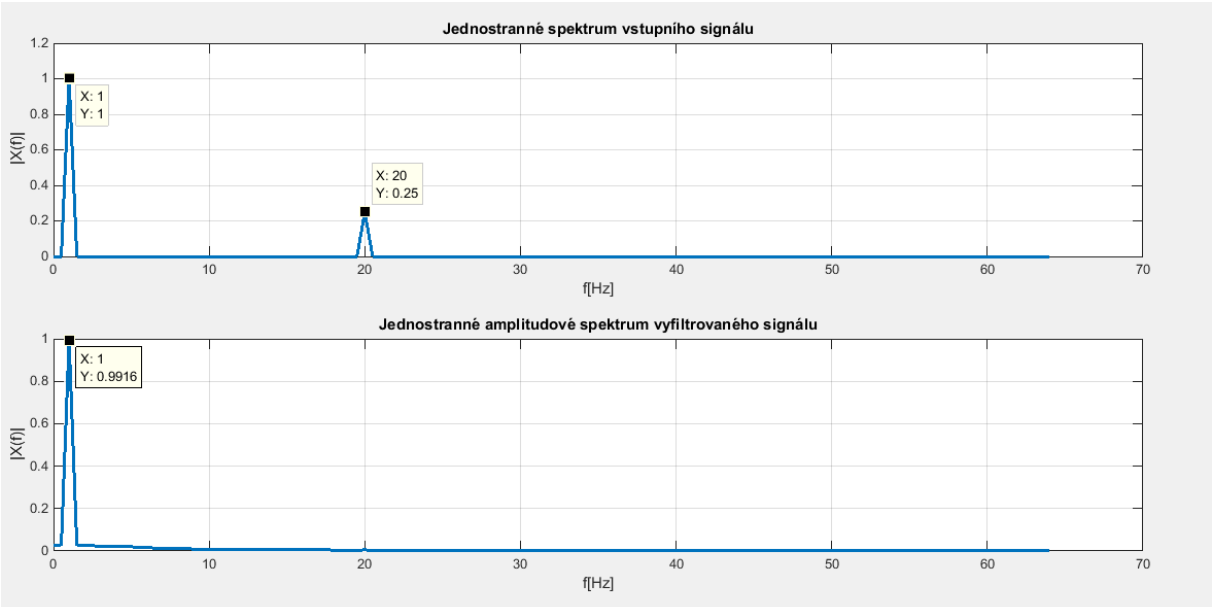
Jak je vidět na obrázku 5.4, všechny póly, značeny křížkem, jsou uvnitř jednotkové kružnice. Zároveň platí, že každý reálný filtr musí mít všechno póly i nulové body na reálné ose nebo musí být podél reálné osy komplexně sdruženy. Čtyři póly přenosové funkce jsou vykresleny v pravé části kružnice a jsou komplexně sdruženy. Je tedy ověřena stabilita signálu.

Na obrázku 5.5 je znázorněna frekvenční charakteristika IIR filtru. Při útlumu -3dB je frekvence 0,039Hz.



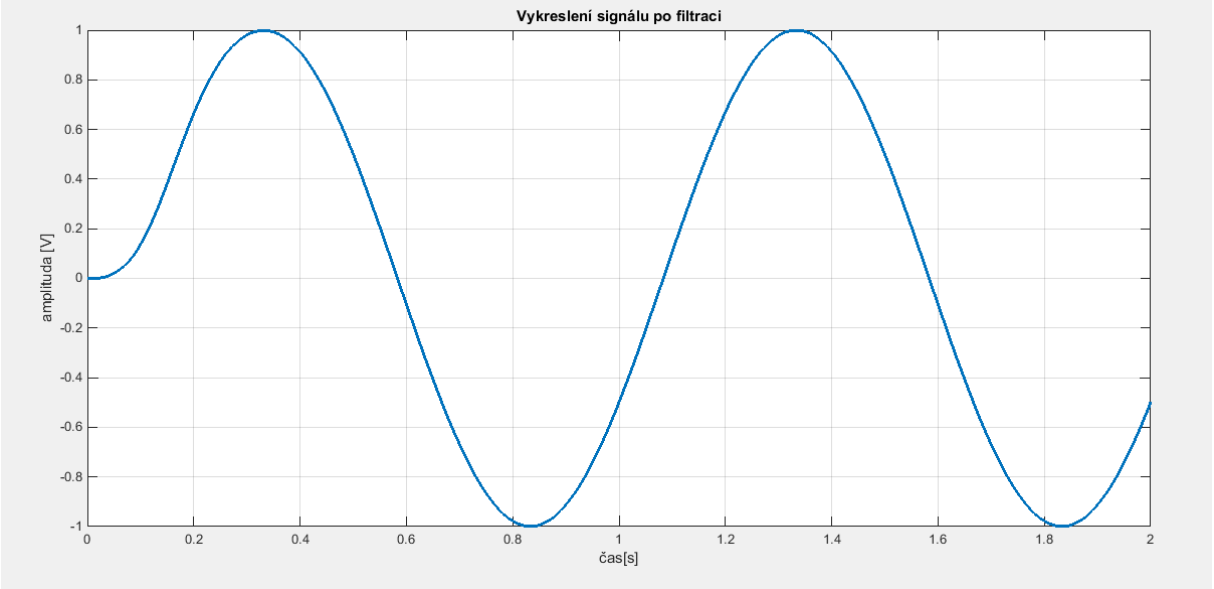
Obrázek 5.5: Amplitudová frekvenční charakteristika

Na obrázku 5.6 lze vidět jednostranné spektrum vstupního signálu (nahoře) a výstupního signálu (dole). Jak můžeme vidět, nežádoucí šum byl vyfiltrován.



Obrázek 5.6: Jednostranné spektrum vstupního signálu

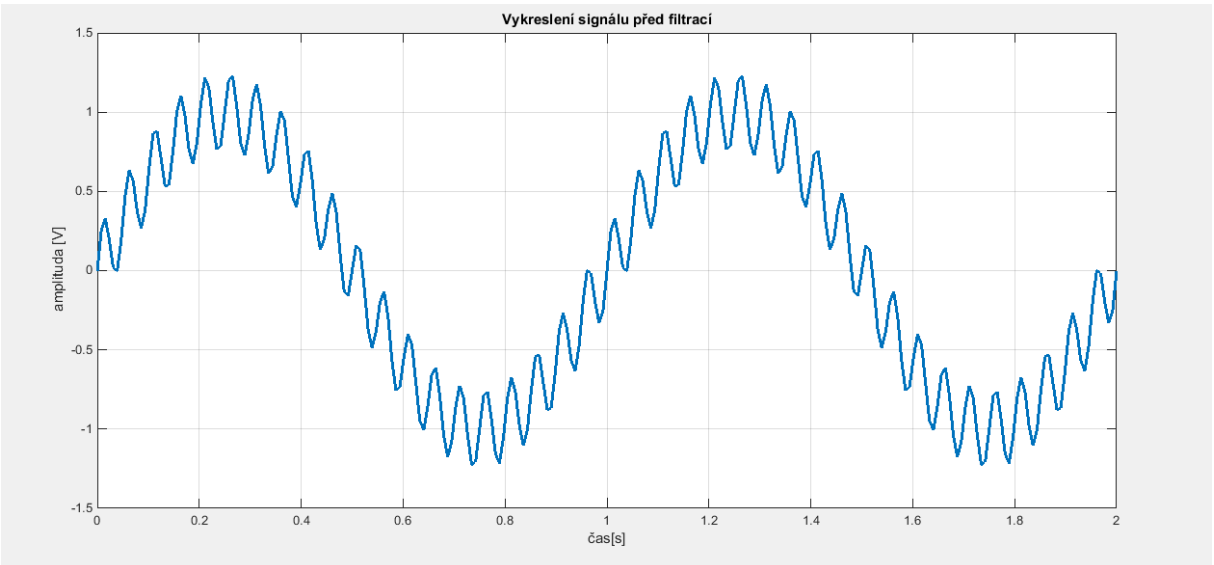
Obrázek 5.7 znázorňuje, že při filtrování signálu pomocí IIR filtrů nám vystačí řád filtru N=4, abychom bylo dosaženo vyfiltrovaného signálu.



Obrázek 5.7: Výstupní signál

5.2 Návrh FIR filtru

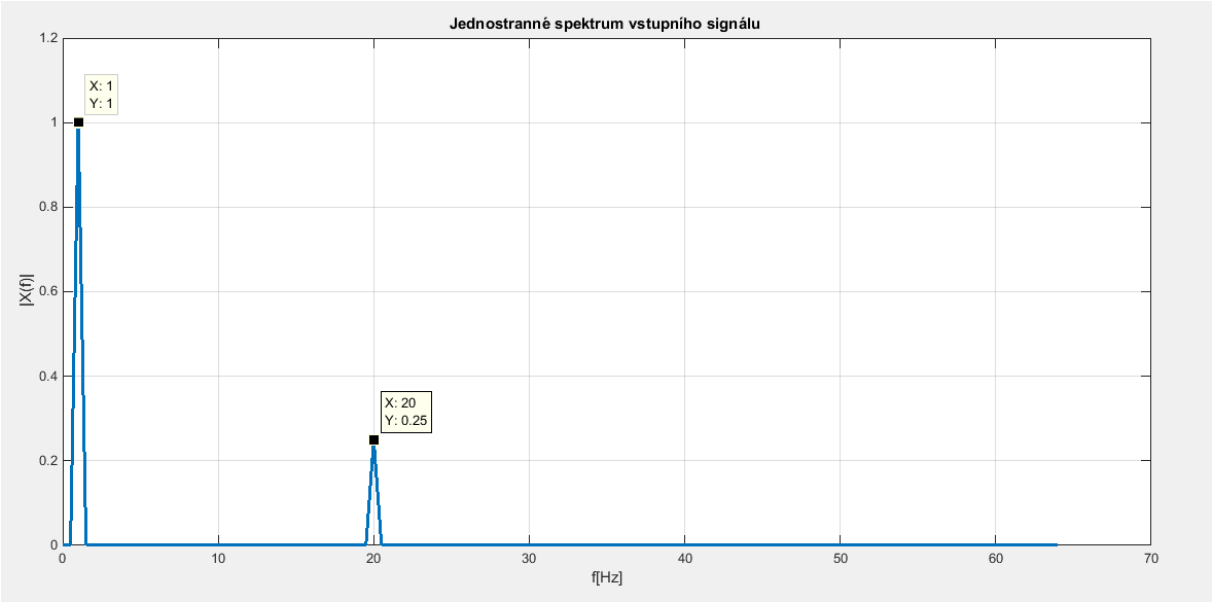
Při návrhu FIR filtru byl zvolen filtr typu dolní propust a stejný signál jako u IIR návrhu $1 \cdot sin(2 \cdot \pi \cdot t) + 0,25 \cdot sin(2 \cdot \pi \cdot 20 \cdot t)$, vzorkovací frekvence byla zvolena 128Hz, a cutoff frekvence 5Hz. Na následujícím obrázku můžeme vidět původní vstupní signál.



Obrázek 5.8: Vstupní signál

Na obrázku 5.9 je zobrazeno jednostranné amplitudové spektrum vstupního signálu. Z něj vyplývá, že kromě sinusovky o amplitudě 1V a frekvenci 1Hz je v signálu obsažen i sinusovka

o amplitudě 0,25V a frekvenci 20Hz, který představuje šum (dále jen šum), který je potřeba ze signálu vyfiltrovat.



Obrázek 5.9: Jednostranné amplitudové spektrum

Pro návrh číslicového FIR filtru typu dolní propust byla vybrána metoda váhových oken, která byla již podrobně charakterizována v podkapitole 4.2.1.1. Díky své jednoduchosti a účinnosti se tato metoda využívá nejčastěji. FIR filtr byl navržen pomocí Hammingova okna a Obdélníkového okna. Pro tento návrh byla zvolena vzorkovací frekvence $F_{vz} = 128$Hz a cutoff frekvence 5Hz.

Při návrhu dolní propusti se určí ideální impulzní odezva dolní propusti, která se vypočítá podle následujícího vzorce:

$$h_d(n) = \{\frac{sin[\omega_c(n-M)]}{\pi(n-M)}, \tag{5.1}$$

kde

M je index středního koeficientu a vypočítá se následovně

$$M = \frac{N}{2}$$

Řády filtru pro obě okna byly zvoleny dva, a to N = 32 a N = 64 pro lepší znázornění FIR filtrů a jeho vlastností.

Obdélníkové okno

Obdélníkové okno je nejjednodušší okno, má však ze všech funkcí oken nejhorší vlastnosti. Všechny koeficienty obdélníkového okna mají stejnou hodnotu rovnou 1.

$$w(n) = 1; \tag{5.2}$$

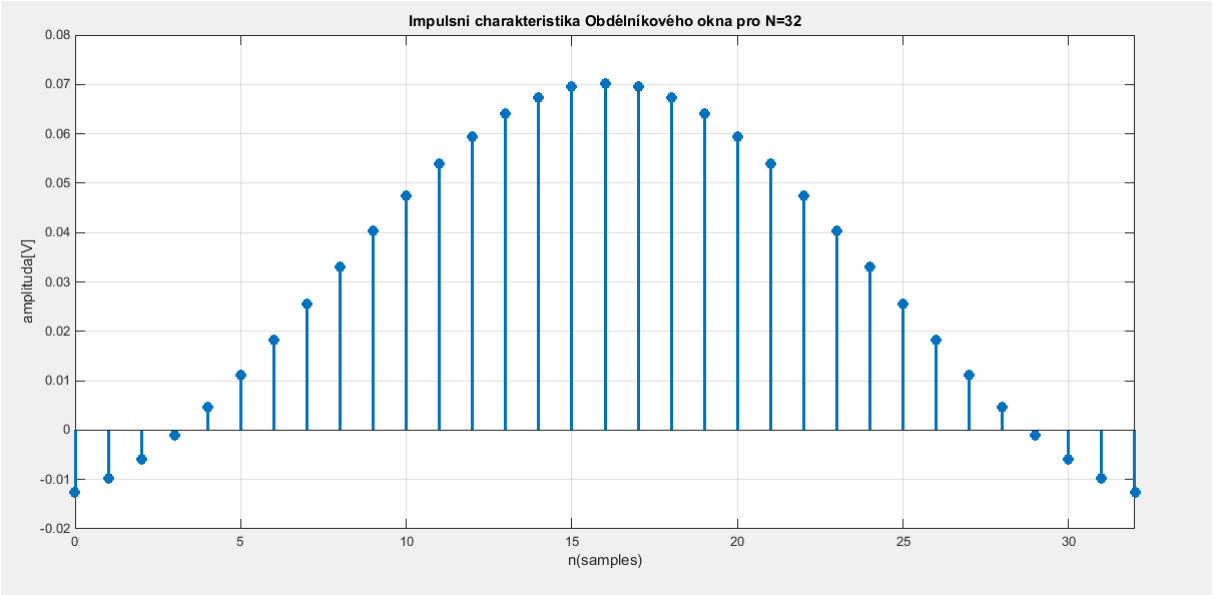
Koeficienty pro návrh FIR filtru jsou získány vynásobením koeficientů oken a koeficientů ideální dolní propusti, jak ukazuje následující vzorec

$$h(n) = h_d(n) \cdot w(n), \tag{5.3}$$

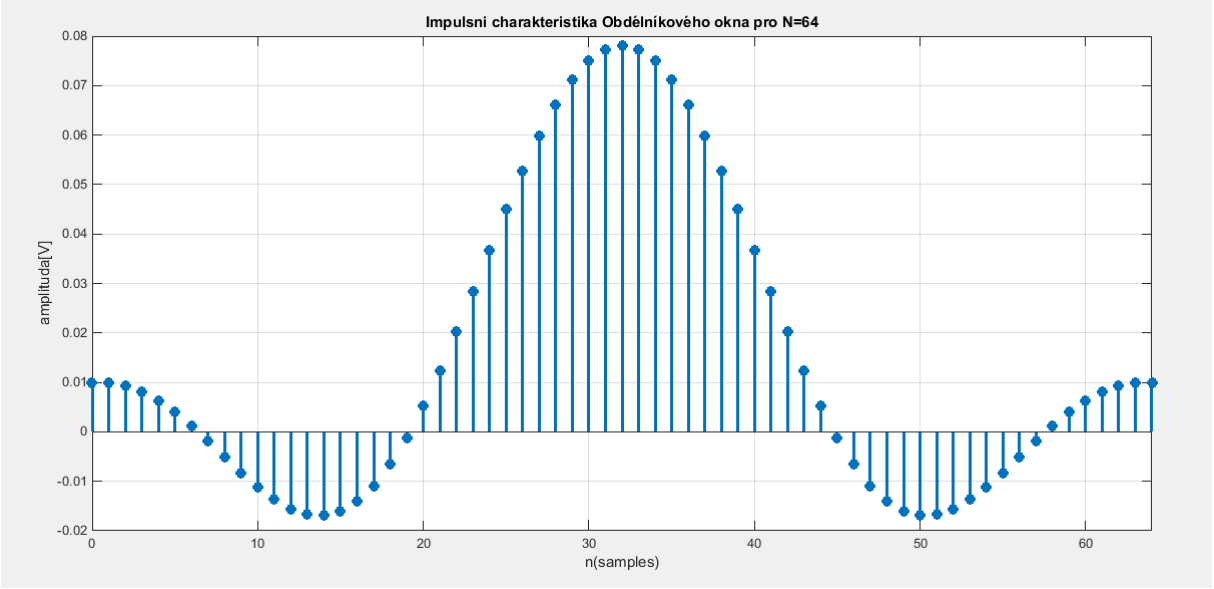
kde

$0 < n <$řád filtru.

Na obrázcích 5.10 a 5.11 lze vidět koeficienty filtru pro řád filtru N=32 a N=64.



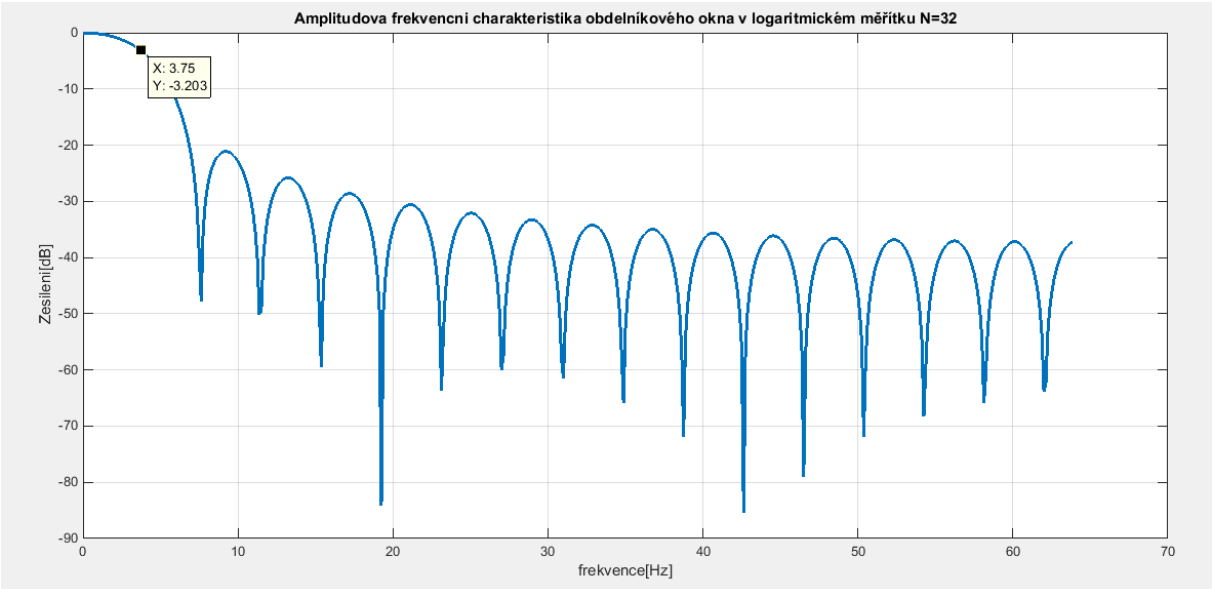
Obrázek 5.10: Impulzní charakteristika koeficientů filtru Obdélníkového okna pro řád filtru N=32



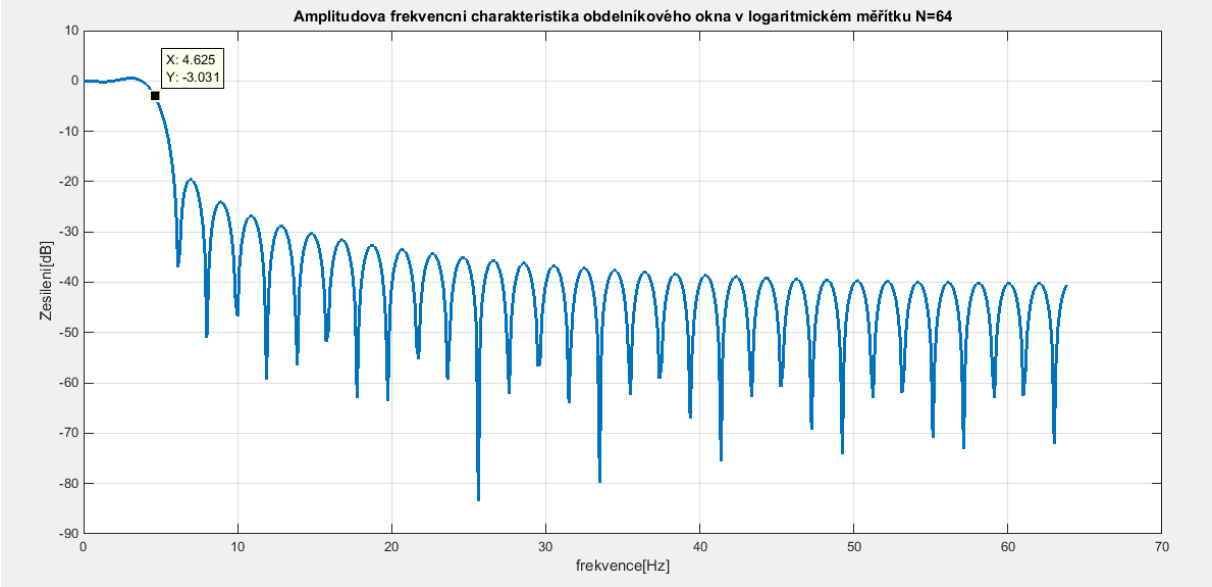
Obrázek 5.11: Impulzní charakteristika koeficientů filtru Obdélníkového okna pro řád filtru N=64

Na obrázcích 5.12 a 5.13 je zobrazena amplitudová frekvenční charakteristika v logaritmickém měřítku Obdélníkového okna řádu filtru N=32 a N=64. Z obrázků lze vidět, že při větším řádu filtru, je lépe vyhlazena kmitočtová charakteristika, ale výpočetní časová náročnost roste.

Při útlumu -3dB je u frekvenční charakteristiky řádu filtru N=32 frekvence 3,75Hz a u řádu filtru N=64 je frekvence 4,625Hz. Je to dáno tím, že při větším řádu filtru má frekvenční charakteristika větší strmost.

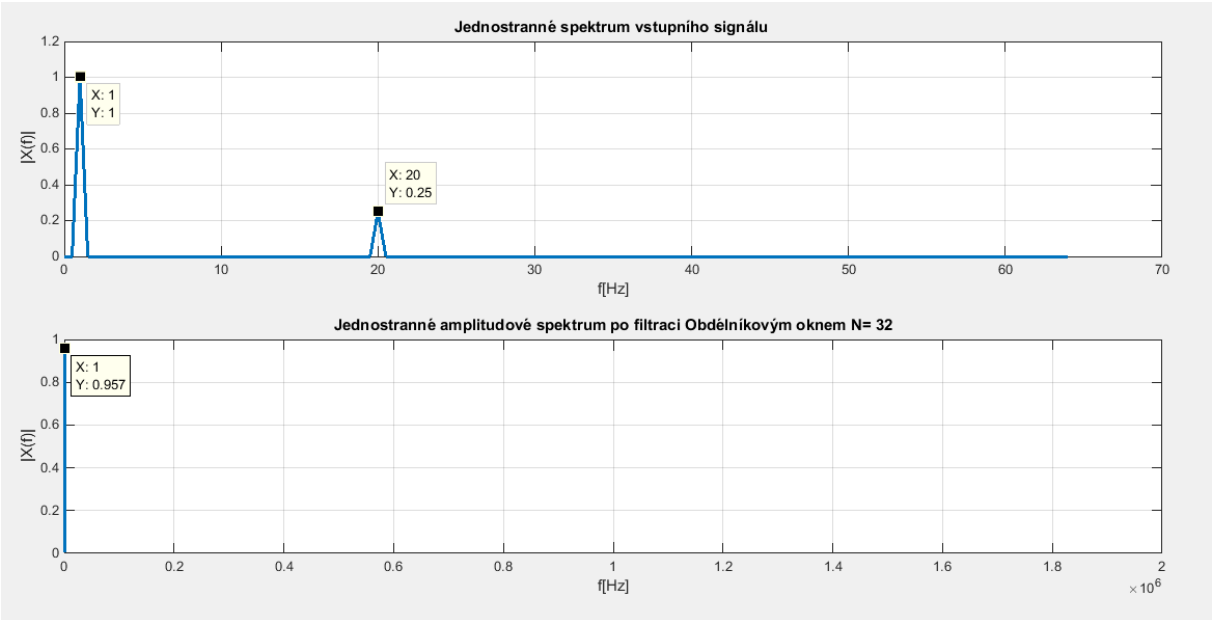


Obrázek 5.12: Amplitudová frekvenční charakteristika v logaritmickém měřítku Obdélníkového okna řádu filtru N=32

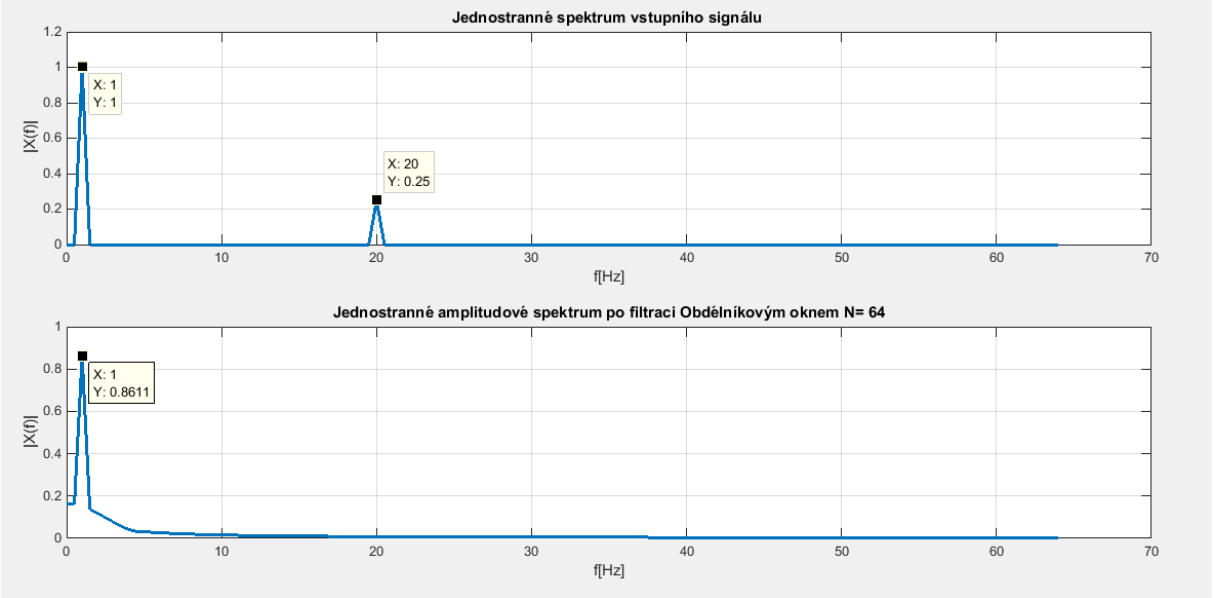


Obrázek 5.13: Amplitudová frekvenční charakteristika v logaritmickém měřítku Obdélníkového okna řádu filtru N=64

Na obrázcích 5.14 a 5.15 je znázorněno jednostranné amplitudové spektrum vstupního signálu a již vyfiltrovaného signálu Hammingovým oknem pro řád filtru N=32 a N=64. Ze spektra můžeme vidět, že došlo k odstranění nežádoucího šumu ze vstupního signálu.

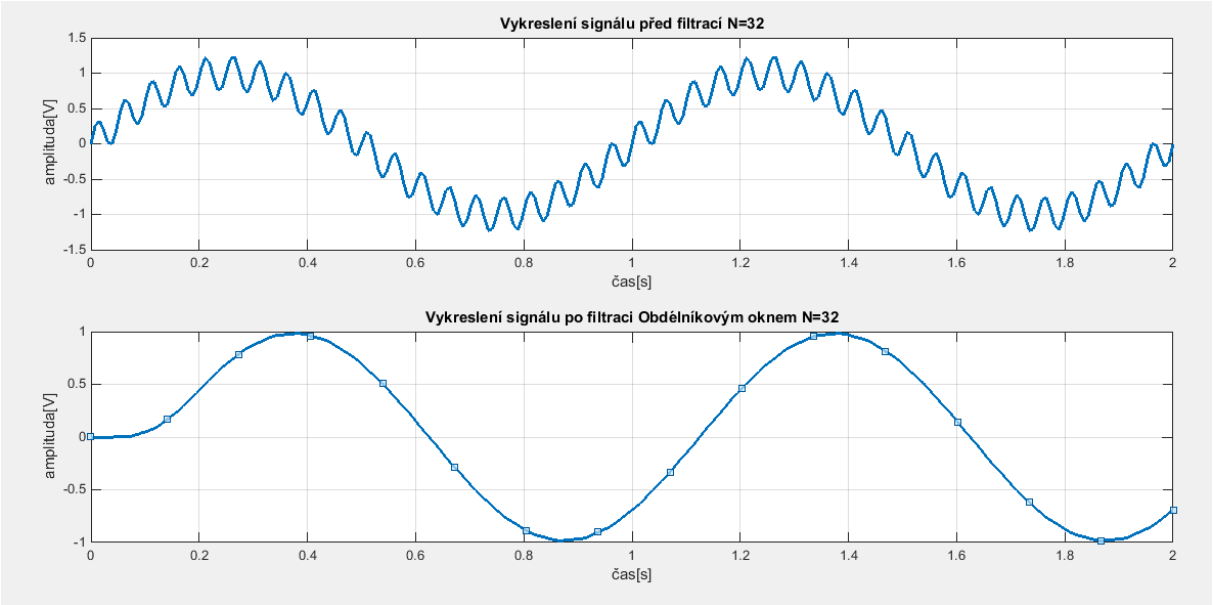


Obrázek 5.14: Jednostranné amplitudové spektrum N=32

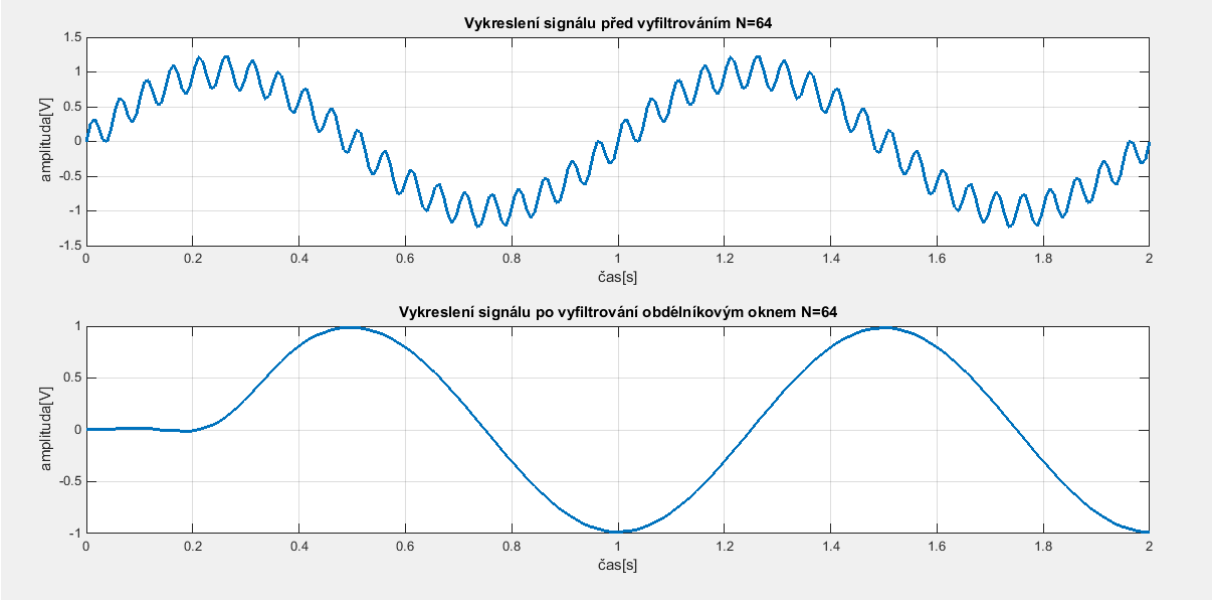


Obrázek 5.15: Jednostranné amplitudové spektrum N=64

Obrázky 5.16 a 5.17 znázorňují nevyfiltrovaný a vyfiltrovaný signál, řád filtru N=32 a N=64.



Obrázek 5.16: Vstupní a výstupní signál, řád filtru N=32



Obrázek 5.17: Vstupní a výstupní signál, řád filtru N=64

Hammingovo okno

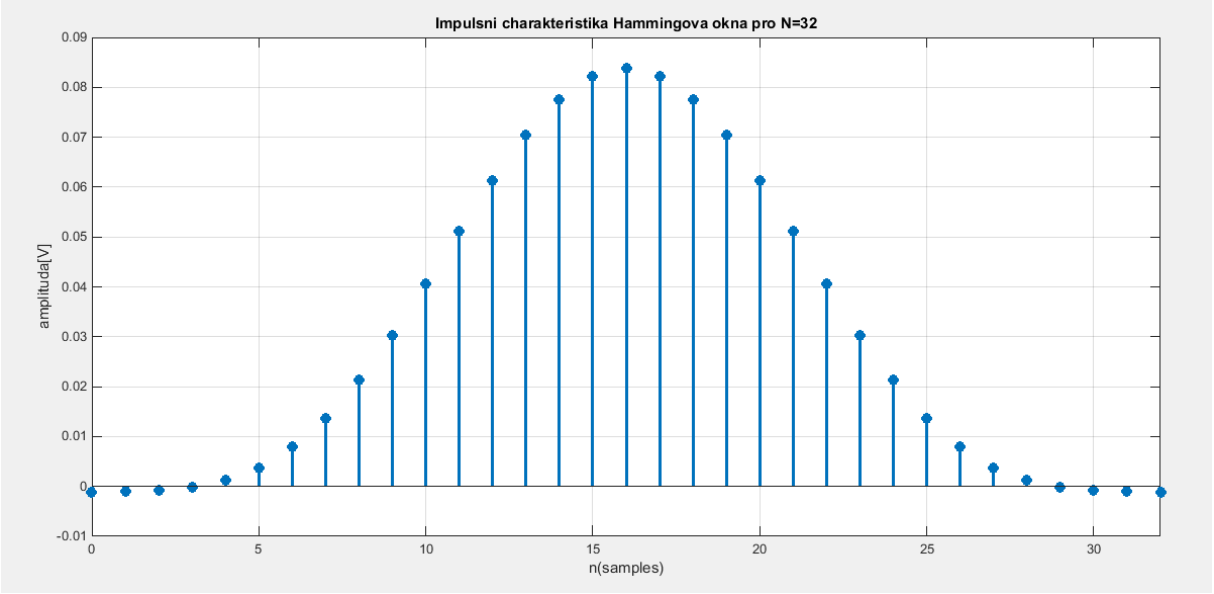
Nejprve musí být vypočítány koeficienty okna, které se vypočítají podle vzorce:

$$w(n) = 0,56 - 0,46(1 - cos\frac{2\pi n}{N-1}), \tag{5.4}$$

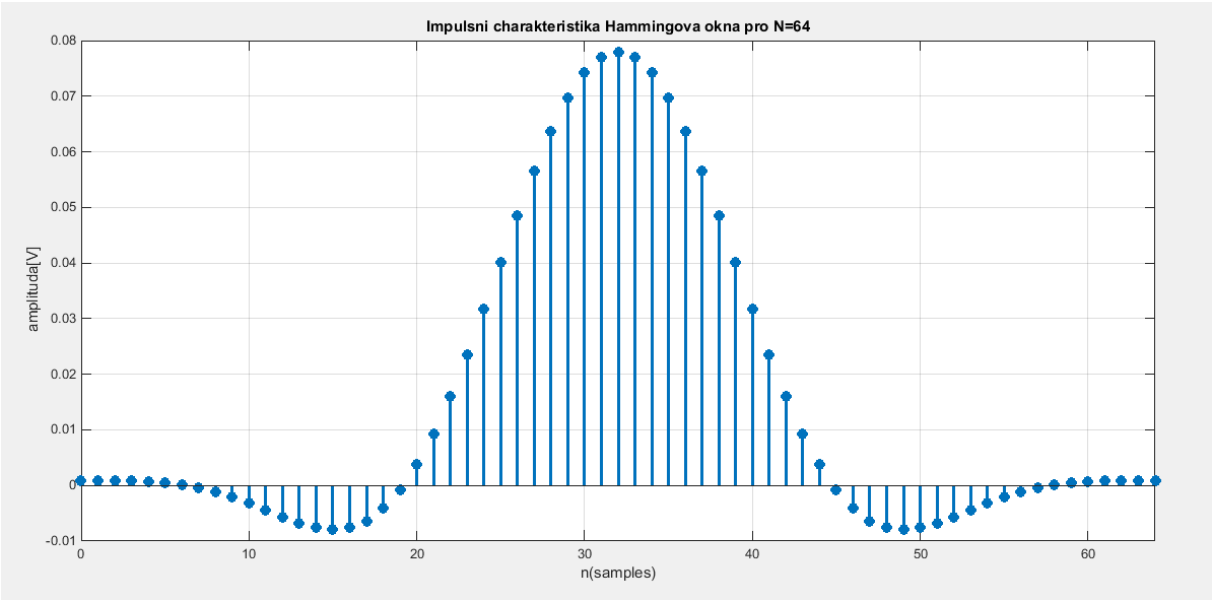
Koeficienty FIR filtru jsou poté nalezeny vynásobením koeficientů okna s koeficienty ideální impulzní odezvy dolní propusti.

$$h(n) = h_d(n) \cdot w(n), \tag{5.5}$$

Impulsní charakteristiku filtru okna pro řád filtru N=32 a N=64 lze vidět na obrázcích 5.18 a 5.19

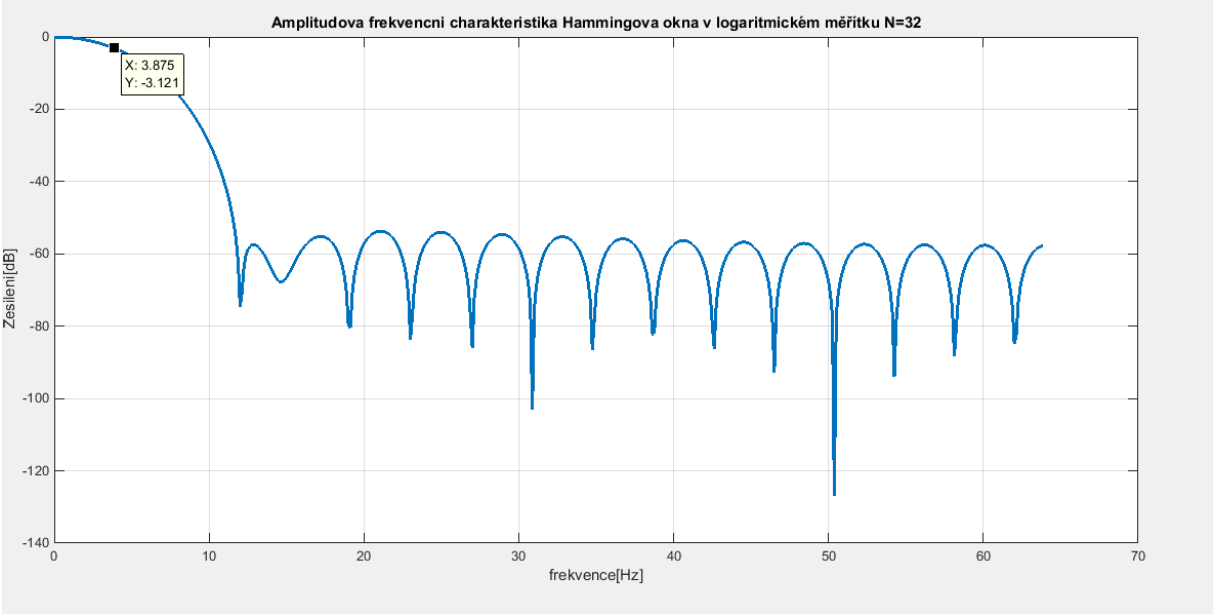


Obrázek 5.18: Koeficienty filtru, Hamminogovo okno, řád filtru N=64

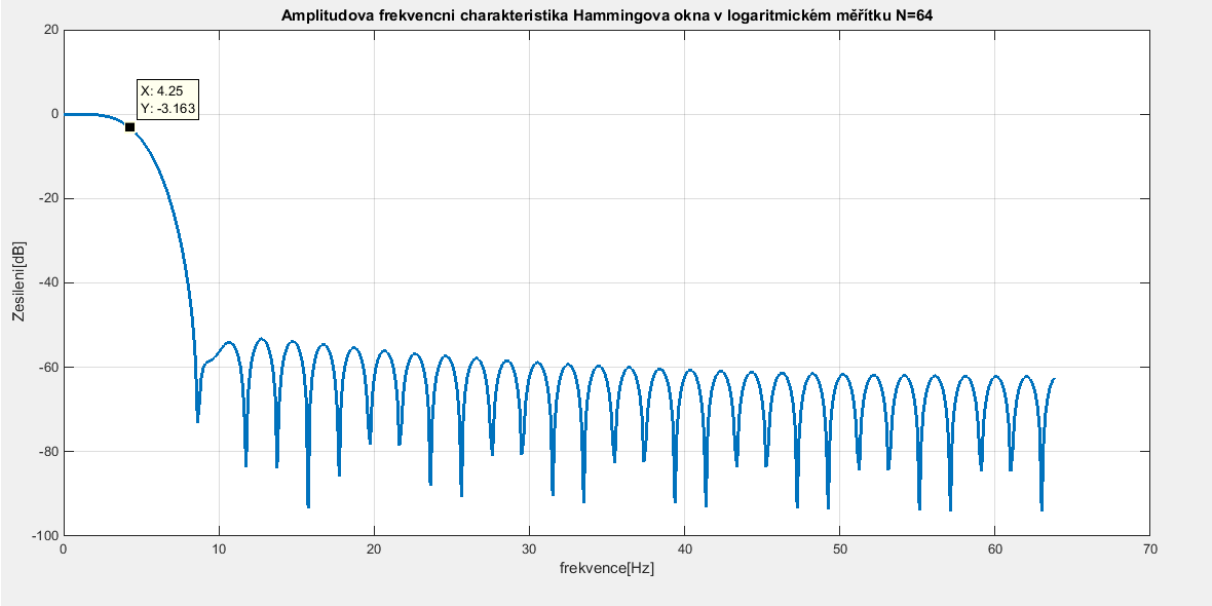


Obrázek 5.19: Koeficienty filtru, Hamminogovo okno, řád filtru N=64

Na obrázku 5.20 a 5.21 je vidět porovnání amplitudových kmitočtových charakteristik Hammingova okna pro N=32 a N=64. Při útlumu -3dB je u řádu filtru N=32 frekvence 3,875Hz u řádu filtru N=64 frekvence 4,25Hz.

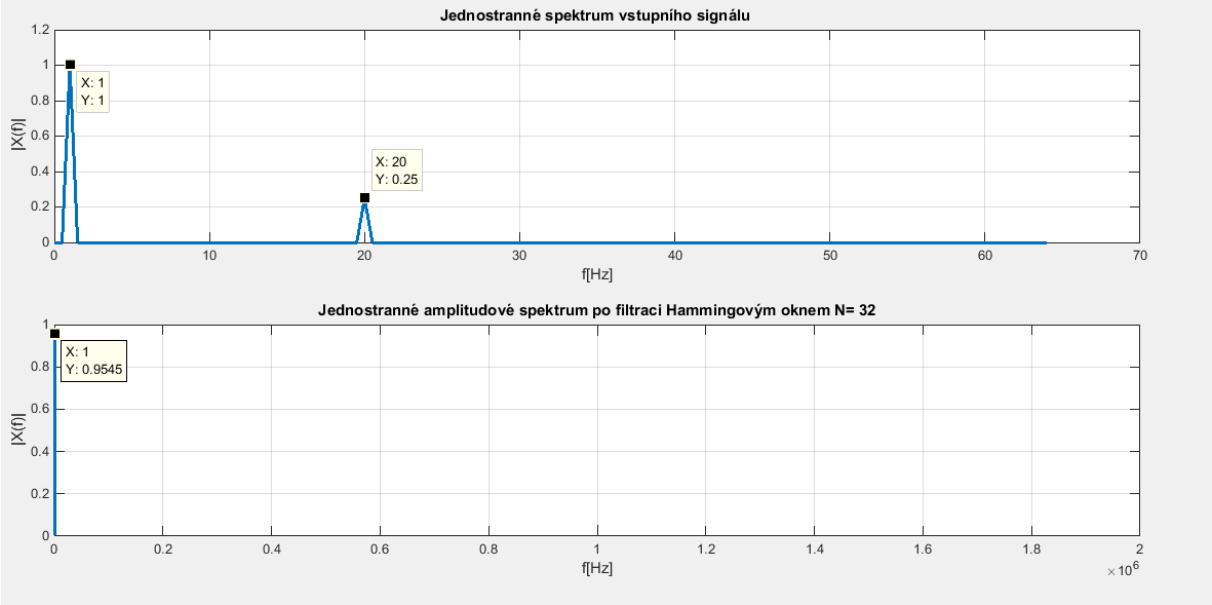


Obrázek 5.20: Amplitudová frekvenční charakteristika Hammingova okna pro řád filtru N=32

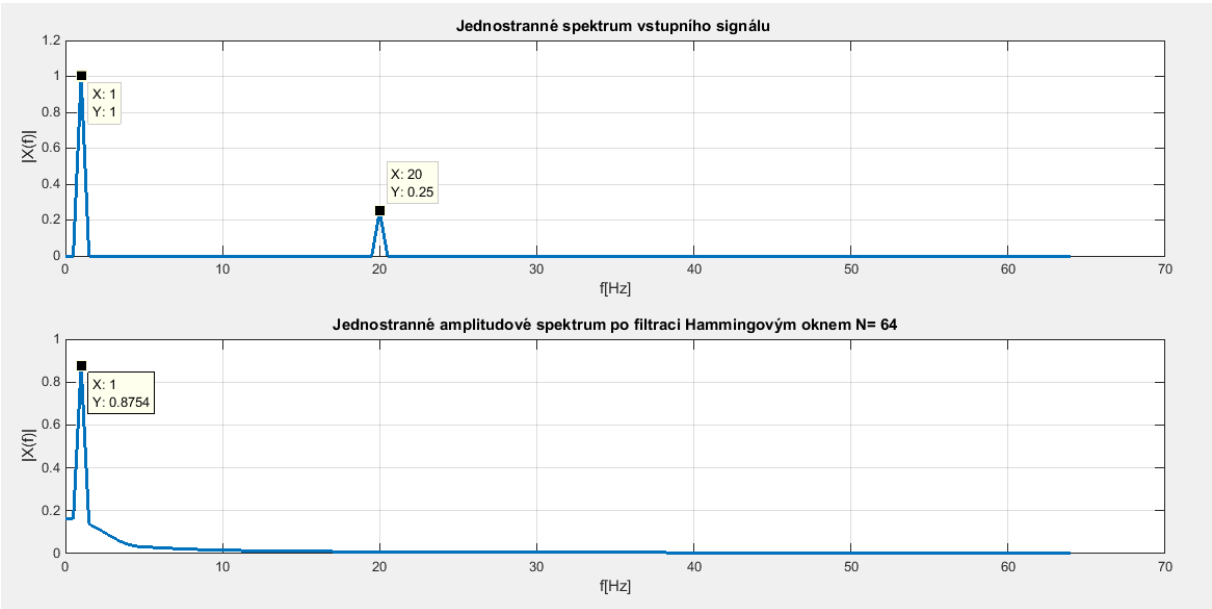


Obrázek 5.21: Amplitudová frekvenční charakteristika Hammingova okna pro řád filtru N=64

Na 5.22 obrázku 5.23 a jsou zaznamenány jednostranné amplitudové spektra vstupního signálu a vyfiltrovaného signálu Hammingovým oknem pro řád filtru N=32 a N=64.

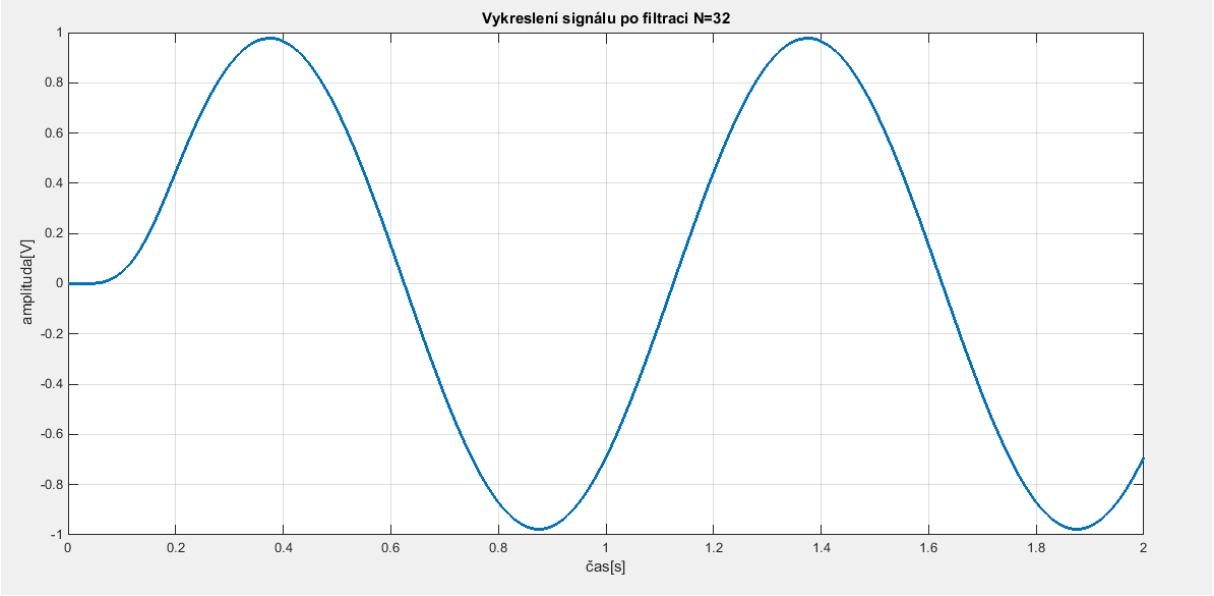


Obrázek 5.22: Jednostranné amplitudové spektrum, Hammingovo okno, N=32

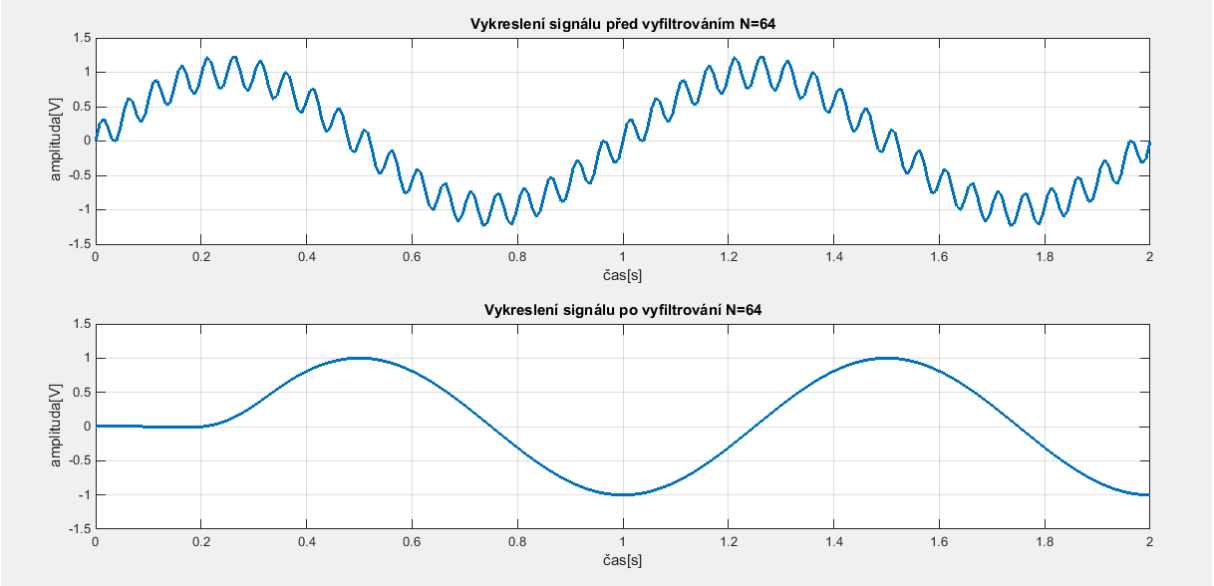


Obrázek 5.23: Jednostranné amplitudové spektrum, Hammingovo okno, N=64

Vyfiltrovaný signál je znázorněn na následujícím obrázku.



Obrázek 5.24: Vyfiltrovaný signál N=32



Obrázek 5.25: Vstupní a výstupní signál, řád filtru N=64

Obrázek 5.25 znázorňuje rozdíl mezi vstupním a vyfiltrovaným signálem.

5.3 Návrh filtru v textovém formátu

K návrhu filtru v prostředí Matlab byly využity funkce, které lze přímo použít k zapsání zdrojového kódu pro návrh filtru. Celý zdrojový kód je k dispozici na přiloženém nosiči CD ROM. V první části zdrojového kódu jsou zadané parametry filtru. *A1* a *A2* jsou amplitudy vstupního

signálu, *f1* a *f2* jsou frekvence vstupního signálu. Proměnná f_{vz} je vzorkovací frekvence a proměnná f_c značí cutoff frekvenci. Samotný návrh FIR filtru probíhá pomocí funkce *fir1* a funkce *filter*. Funkce *fir1* vrací hodnoty koeficientů daného okna a návratovou hodnotou funkce *filter* jsou koeficienty navrženého filtru.

Návrh IIR filtru probíhá pomocí funkce *butter*, která vrací koeficienty A a B, a pomocí funkce *filter*, která vrací koeficienty navrženého filtru. Pomocí funkce *fft* jsou vykreslena jednostranná spektra signálu. Funkce *fft* počítá diskrétní Fourierovu transformaci (DFT) za použití algoritmu rychlé Fourierierovy transformace (FFT). Impulzní charakteristika okna je vykreslena pomocí funkce *impz* a amplitudová frekvenční charakteristika pomocí funkce *freqz*. Funkce *zplane* vrací zakreslení nulových pólů a bodů z přenosové funkce.

6 Implementace FIR a IIR filtru

FIR a IIR filtry byly implementovány na jádře ARM Cortex M0+ na kitu TWR-KM34Z50M programovacím jazyce C v programu IAR Embedded Workbench s použitím a bez použití matematického koprocesoru.

6.1 MMAU

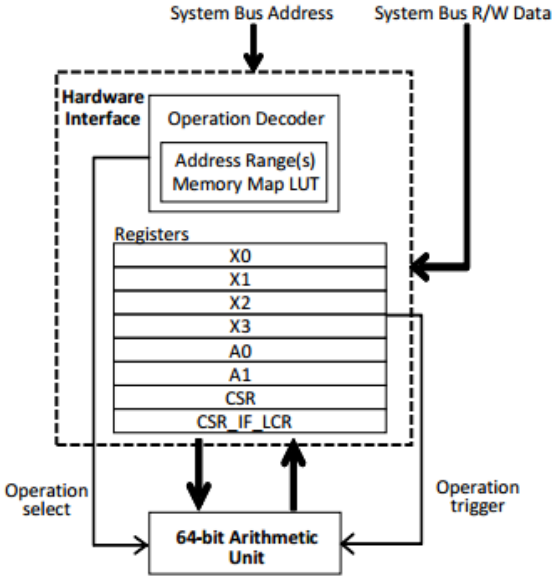
Je 64-bitový matematické koprocesor.

Běžně používané mikroprocesory používané pro aplikace na měření elektrické energie mají integrovaný sigma-delta A/D převodník s 24-bitovým nebo dokonce vyšším dynamickým rozsahem měření. Aby bylo využito této výhody měření s vysokým rozlišením, musí být jejich zpracování prováděno s nejméně 24-bitovou přesností.

Standardní procesorová jádra, včetně Cortex-M0 +, jsou optimalizována pro všeobecné použití aplikací, které nemají hardwarovou podporu pro 32/64-bitovou přesnost měření, druhou odmocninu a MAC operací. Tyto instrukce lze emulovat v softwaru, nicméně, emulace softwaru není možná v mnoha případech kvůli runtime omezení.

Vnitřní struktura MMAU je znázorněna na obrázku 6.1. Paměťově mapované zařízení je umístěno na portu systémové sběrnice, paměťové adresy MMAU reagují na jeho programovací model.

Mezi hardwarové prvky MMAU patří: hardwarové rozhraní, dekodér operací s registry a aritmetická jednotka.



Obrázek 6.1: Vnitřní struktura MMAU

Výkon samostatné aritmetické jednotky může poskytnout několikanásobné zrychlení pro výpočty s vysokým dynamickým rozsahem oproti nejběžnějším mikrokontrolérům. S cílem maximalizovat výpočetní výkon bylo hardwarové rozhraní MMAU založeno na principu paměťového mapovaní výpočtu instrukcí.

Výhodou je, že provádění instrukce se spouští v rámci hardwarového rozhraní jediným přístupem do paměti, ten je nutný k načtení operandu do vstupního registru a vyvolání požadované aritmetické operace.

MMAU umožňuje vývoj pomocí jednoduchých, krátkých a velmi účinných softwarových balíčků pro načítání operandů a získávání výsledků z aritmetické jednotky.

Matematický koprocesor podporuje datový typy unsigned, signed integer - ty jsou vhodné pro všeobecné počítání, a datový typ fraction - ten je vhodný pro algoritmy na zpracování digitálního signálu.

Datový typ fractional

Tento datový typ má floatový rozsah $< -1, 1 >$, fractional rozsah $< -2^{(n-1)}, 2^{(n-1)} >$ kde n je 16,24,32 nebo 64 podle typu FRAC.

Mezi tento datový typ patří FRAC16, FRAC24, FRAC32 a FRAC64 - číslo znázorňuje, s jakou bitovou přesností pracuje. Může tedy počítat na přesnost až 64 bitů.

Převod tohoto datového typu Frac na float se dá vypočítat podle následujícího vzorce

$$float = \frac{y}{2^n},$$

kde

y = číslo v datovém typu FRAC

n = 31, 63 - podle toho, z jakého datového typu se převádí.

Softwarové ovladače v současné době obsahují 160 softwarových funkcí tak, aby uživatelům poskytly plný přístup k MMAU saturačním a nesaturačním operacím, které mohou být převedené unsigned, signed integer nebo do fractional datových typů.

Instrukce MMAU

Datové typy signed a unsigned integer a fractional Q0.31 nebo Q0.63 podporují tyto operace:

- 32x32 nebo 64x32 násobení
- 32x32 nebo 64x32 operace MAC(multiplicate and acumulate)
- 32/32 nebo 64/32 dělení
- 32 a 64 odmocniny

Aritmetická jednotka počítače násobí a provádí operaci MAC v jednom cyklu. Instrukce jsou dodávány v knihovně *fraclib* a dělí se podle:

- Typu operace - operace MAC, násobení, dělení a odmocnina
- Datového typu - unsigned integer, signed integer, fractional
- Návratové /bez návratové hodnoty (mezivýsledek zůstává uložený ve vnitřním akumulátoru MMAU.)

V tabulkách jsou zaznamenány instrukce pro datový typ FRAC32 a FRAC64.

Tabulka 3: Instrukce pro datový typ FRAC32

FRAC32	**MAC**	**Násobení**	**Dělení**	**Odmocnina**
Návratová hodnota	l_mac_ll l_mac_dl	l_mul_ll l_mul_dl	l_div_ll	l_sqr_l l_sqr_d
Bez návratové hodnoty	mac_ll mac_dl	mul_ll mul_dl	div_ll div_dl	sqr_l sqr_d

Tabulka 4: Instrukce pro datový typ FRAC64

FRAC64	**MAC**	**Násobení**	**Dělení**	**Odmocnina**
Návratová hodnota	d_mac_ll d_mac_dl	d_mul_ll d_mul_dl	d_divll	- -
Bez návratové hodnoty	mac_ll mac_dl	mul_ll mul_dl	div_ll div_dl	sqr_l sqr_d

6.2 Implementace IIR filtru

Při implementaci IIR filtru byly hodnoty signálu vygenerované Matlabem, a poté převedeny na datový typ FRAC64. Koeficienty IIR filtru byly rovněž vygenerované Matlabem a převedeny na datový typ FRAC32. IIR filtry byly implementovány s použitím MMAU bez návratové hodnoty, s MMAU s návratovou hodnotou a bez MMAU. Řád filtru byl zvolen N=4.

IIR filtry s použitím MMAU bez návratové hodnoty

```
static frac64 d_iir_4ord_mmau (frac64 x, const frac32 *pb, const frac32 *pa,
    int16 sc, frac64 *px, frac64 *py)
{
  register frac64 y;

  /* Aktualni vypocet vystupni hodnoty filtru s vyuzitim instrukci MMAU */
```

```
  mul_dl(x, pb[0] );
  mac_dl(px[0],pb[1]);
  mac_dl(px[1],pb[2]);
  mac_dl(px[2],pb[3]);
  mac_dl(px[3],pb[4]);
  mac_dl(py[0],pa[0]);
  mac_dl(py[1],pa[1]);
  mac_dl(py[2],pa[2]);
  y = d_mac_dl(py[3],pa[3])<<sc;

  /* posun predchozich vstupnich hodnot                                    */
  px[3]=px[2]; px[2]=px[1]; px[1]=px[0]; px[0]= x;

  /* posun predchozich vystupnich hodnot                                   */
  py[3]=py[2]; py[2]=py[1]; py[1]=py[0]; py[0]= y;

  return y;

}
```

Výpis 1: IIR filtry s použitím MMAU

Při implementaci byly použity instrukce mac_dl (MAC operace 32 a 64 bitových čísel), mul_dl (násobení 64 a 32 bitových čísel) a d_mac_dl(MAC operace, která vrací 64 bitový obsah akumulátoru).

IIR filtry bez použití MMAU

Zdrojový kód je stejný jak pro implementaci IIR filtru s použitím MMAU s návratovou hodnotou i bez použití MMAU. V knihovně *FRACLIB_Inlines.h* se pouze povolí nebo zakáže používání MMAU příkazem $\#define\ USE\_MMAU$

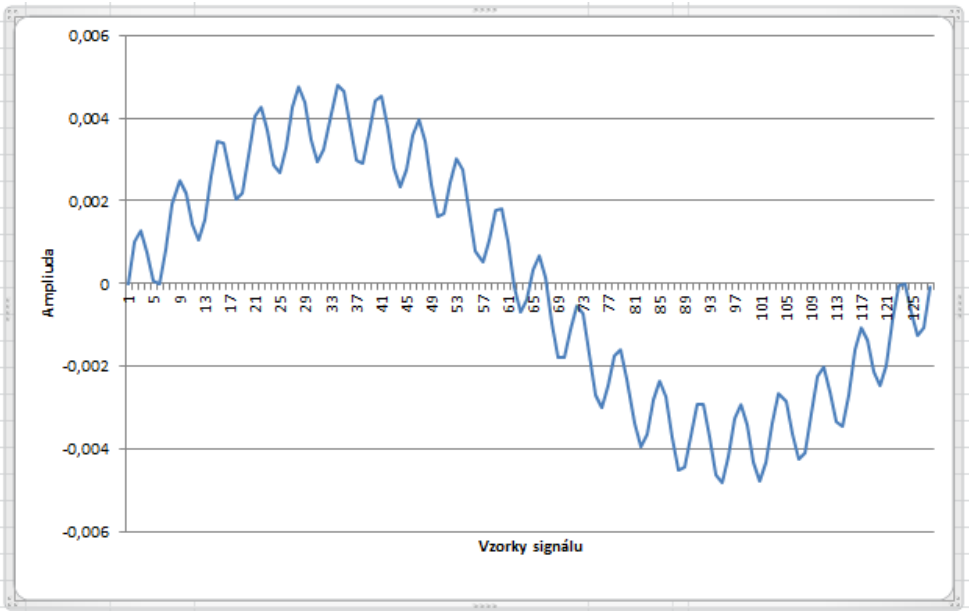
```
static frac64 d_iir_4ord (frac64 x, const frac32 *pb, const frac32 *pa, int16
    sc,
                          frac64 *px, frac64 *py)
{
  register frac64 y;

  /* Aktualni vypocet vystupni hodnoty filtru s vyuzitim instrukci MMAU */
  y = LL_mul_lll(x, pb[0] );
  y = LL_mac_lll(px[0],pb[1]);
  y = LL_mac_lll(px[1],pb[2]);
```

```
  y = LL_mac_lll(px[2],pb[3]);
  y = LL_mac_lll(px[3],pb[4]);
  y = LL_mac_lll(py[0],pa[0]);
  y = LL_mac_lll(py[1],pa[1]);
  y = LL_mac_lll(py[2],pa[2]);
  y = LL_mac_lll(py[3],pa[3])<<sc;

  /* posun predchozich vstupnich hodnot                                    */
  px[3]=px[2]; px[2]=px[1]; px[1]=px[0]; px[0]= x;

  /* posun predchozich vystupnich hodnot                                     */
  py[3]=py[2]; py[2]=py[1]; py[1]=py[0]; py[0]= y;

  return y;

}
```

Výpis 2: IIR filtry bez použití MMAU

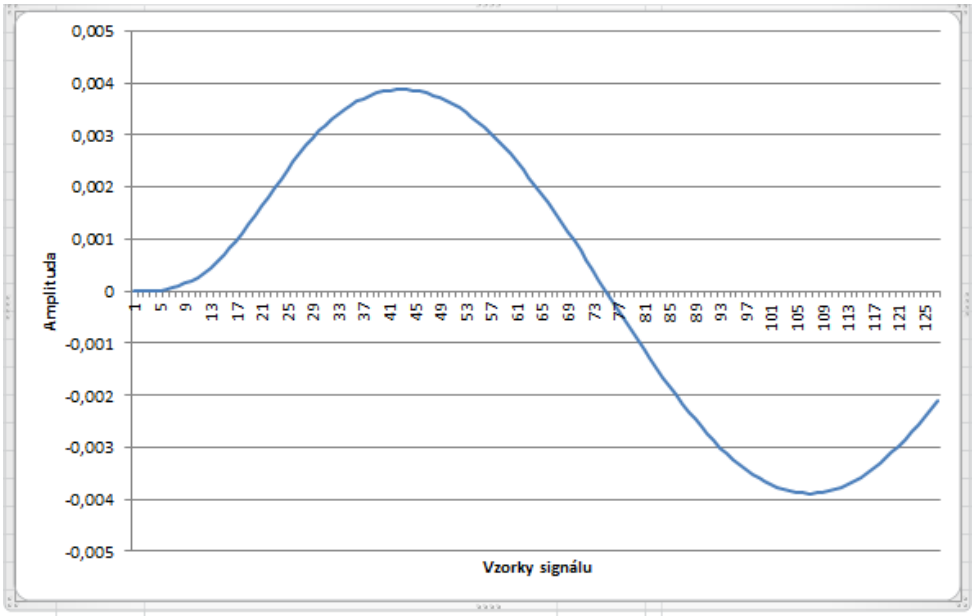
Zdrojový kód používá tyto instrukce LL_mul_lll, LL_mac_lll. Všechny mezivýpočty ukládá do proměnné y.

Na obrázku 6.2 je zobrazen původní signál vykreslený v programu Microsoft Excel.



Obrázek 6.2: Vstupní signál IIR filtru

Po vyfiltrování signálu pomocí IIR filtru dostáváme plně vyfiltrovaný signál, který je zobrazen na obrázku 6.3



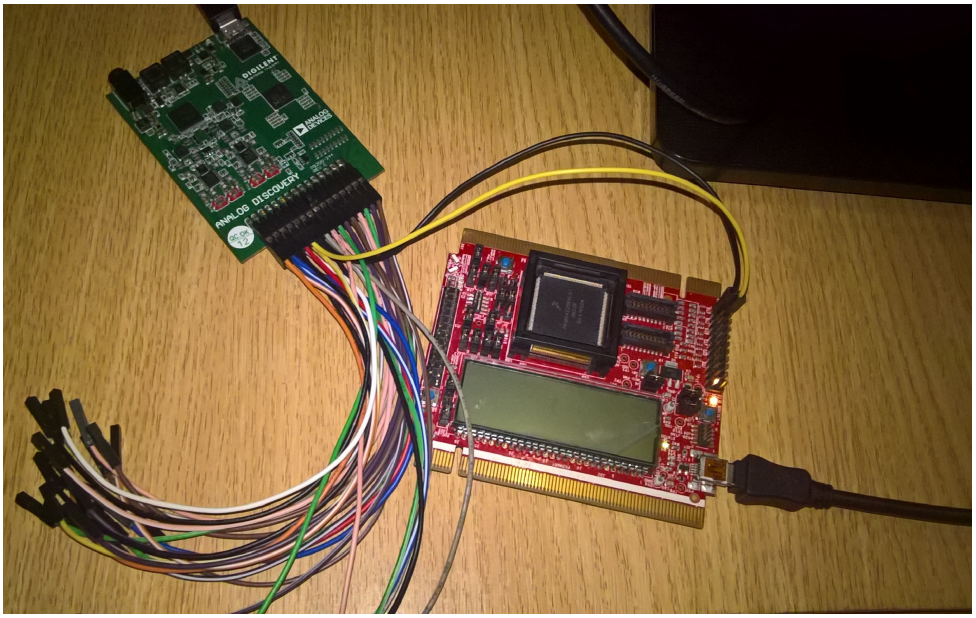
Obrázek 6.3: Výstupní signál IIR filtru

6.3 Implementace FIR filtru

Při implementaci FIR filtru byla použita metoda váhových oken, řád filtru byl použít N=8, N=32 a N=64. Koeficienty filtru byly vypočítány programem MATLAB a převedeny na datový typ FRAC32. Implementace FIR byla s použitím MMAU bez návratové hodnoty, MMAU s návratovou hodnotou a bez použití MMAU.

Při implementaci byl použit sigma-delta A/D převodník. V programu WaveForms byl zvolen signál, který je potřeba vyfiltrovat, pomocí signálového generátoru AnalogDiscovery byl tento signál přiveden na kit, kde sigma-delta A/D převodník převedl tento analogový signál na digitální.

Na následujícím obrázku je zobrazeno zapojení kitu se signálovým generátorem AnalogDiscovery.



Obrázek 6.4: Zapojení kitu se signálovým generátorem

Nastavení sigma-delta A/D převodníku

```
/* Nastaveni frekvence jednotlivych casti cipu (jadro, sbernice, FLASH pamet)*/
SIM_SetClkMode (SYSCLK_MODE1);
SIM_SetClkDiv (SYSCLK_DIV1);
FLL_Init (FLL_MODULE_FEE_48MHZ_CONFIG);

/* zapnuti PLL, jako zdroj krystal */
PLL_Enable (PLL32KREF_SRC1);
```

Pro AFE (sigma delta převodník) si lze vybrat jiný zdroj hodin, než má jádro, sběrnice a FLASH paměť. Je to z důvodu, že do AFE může jít maximální frekvence 6,5 MHz. Jádro není na přesnost tak citlivé, ale AFE ano. Jako zdroj frekvence pro AFE tzn. PLL (fázový závěs), ten násobí vstupní frekvenci poměrem 375. Jako vstupní frekvence se vybere 32,768 kHz krystal, který je velmi přesný.

```
/* Nastaveni vnitrni napetove reference */
VREF_Init (VREF_MODULE_CHOP_ON_1V75_0V4_HPWR_CONFIG,
VREF_SWITCH_S1_L_S2_L_S3_L_BUFF_ON);

/*zvoleni podskupiny zdroju frekvenci PPL */
SIM_SelAfePllClkSrc (SIM_MCG_PLL_CLK);

/* Nastaveni kanalu */
```

```
AFE_ChanInit (CH0, AFE_CH_SWTRG_CCM_PGAOFF_CONFIG(DEC_OSR2048), 0, PRI_LVL0,
    NULL);
AFE_Init (AFE_MODULE_LJFORMAT_CONFIG(AFE_PLL_CLK, AFE_DIV2, 12288000));
```

Při nastavení kanálu se zvolí: software trigger - (každé měření se spustí příkazem), PGA se vypne (nezesiluje se dál signál), CCM - continuous mode (po odstartování měření se měří pořád), DEC-OSR2048 (A/D převodník udělá 2048 jednobitových měření, které se zprůměrují do jednoho naměřeného vzorku).

```
/* Zapnuti mereni */
AFE_SwTrigger (CH0);
```

Při implementaci FIR filtru byl použít algoritmus kruhového bufferu.

Kruhový buffer se skládá ze dvou ukazatelů a z pole, které má délku, který je roven řádu filtru. První ukazatel míří na první obsazený prvek, druhý na první volné místo. Když je do bufferu přidán nový prvek, druhý ukazatel se zinkrementuje. Pokud dojde odstranění prvního prvku z fronty, zinkrementuje se první ukazatel. Obě inkrementace jsou prováděny modulárně, tj. pokud je přidán prvek a konec pole je obsazen, tak se prvek přidá na uvolněný začátek pole. Tak struktura buffer tvoří kruh.

Ve zdrojovém kódu je zobrazena funkce FIR filtru s použitím MMAU.

```
static frac64 fir_circular_buffer_mmau(const frac32 * coef, frac64 input,
    volatile frac64 * output, unsigned int volatile * pointer, unsigned int
    volatile * pointer_out, unsigned int rad_filtru)
{

    frac64 out;
    int pointer_inter;
    int pointer_p = *pointer;
    int pointer_out_p = *pointer_out;
    circular_bufferM[pointer_p++] = input;
    pointer_inter = pointer_p;

    if(pointer_p == rad_filtru){
      pointer_p = 0;
      pointer_inter = 0;
    }

    int i = 0;
    mul_dl(circular_bufferM[pointer_inter++], coef[i]);
```

```
    //Pri naplneni bufferu dojde k resetu indexu bufferu
    if(pointer_inter == rad_filtru){
       pointer_inter = 0;
    }

     //Filtrovani signalu
    for(i=1; i<rad_filtru-1; i++){
      mac_dl(circular_bufferM[pointer_inter++], coef[i]);

      if(pointer_inter == rad_filtru){
          pointer_inter = 0;
      }
    }
    //Zapis vysledku do vystupu
    out = d_mac_dl(circular_bufferM[pointer_inter++], coef[rad_filtru]);

    if(pointer_inter == rad_filtru){
      pointer_inter = 0;
    }

    * pointer = pointer_p;
    * pointer_out = pointer_out_p;
    return out;
}
```

Výpis 3: FIR filtry s použitím MMAU

Stejně jak tomu bylo u IIR filtru, tenhle zdrojový kód je stejný, jako při filtrování signálu pomocí MMAU s návratovou hodnotou. V knihovně *FRACLIB_Inlines.h* se pouze povolí nebo zakáže používání MMAU příkazem.

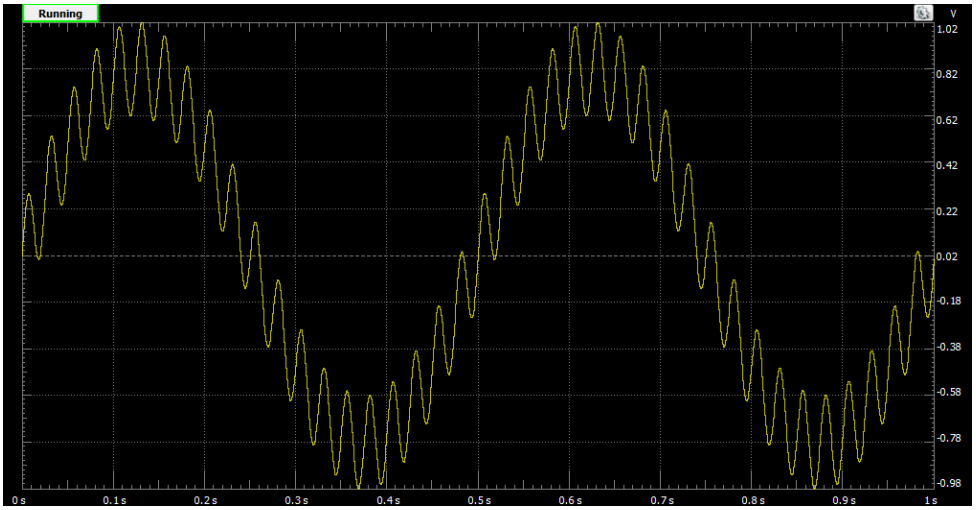
```
static void fir_circular_buffer(const frac32 * coef, frac64 input, volatile
    frac64 * output, unsigned int volatile * pointer, unsigned int volatile *
    pointer_out, unsigned int rad_filtru)
{

    int pointer_inter;              //index bufferu
    int pointer_p = *pointer;
    int pointer_out_p = *pointer_out;
    volatile frac64 acc = 0;
```

```
    circular_bufferM[pointer_p++] = input;
    pointer_inter = pointer_p;

    if(pointer_p == rad_filtru){
      pointer_p = 0;
      pointer_inter = 0;
    }

    int i = 0;
    acc = LL_mul_lll(circular_bufferM[pointer_inter++], coef[i]);

    //Pri naplneni bufferu dojde k resetu indexu bufferu
    if(pointer_inter == rad_filtru){
        pointer_inter = 0;
    }

    //Filtrovani signalu
    for(i=1; i<rad_filtru-1; i++){
      acc = LL_mac_lll(acc,circular_bufferM[pointer_inter++], coef[i]);
      if(pointer_inter == rad_filtru){
          pointer_inter = 0;
      }
    }
    acc = LL_mac_lll(acc,circular_bufferM[pointer_inter++], coef[rad_filtru]);
    if(pointer_inter == rad_filtru){
      pointer_inter = 0;
    }

    //Zapis vysledku do vystupu
    output[pointer_out_p++] = acc;
    * pointer = pointer_p;
    * pointer_out = pointer_out_p;
}
```

Výpis 4: FIR filtry bez použití MMAU

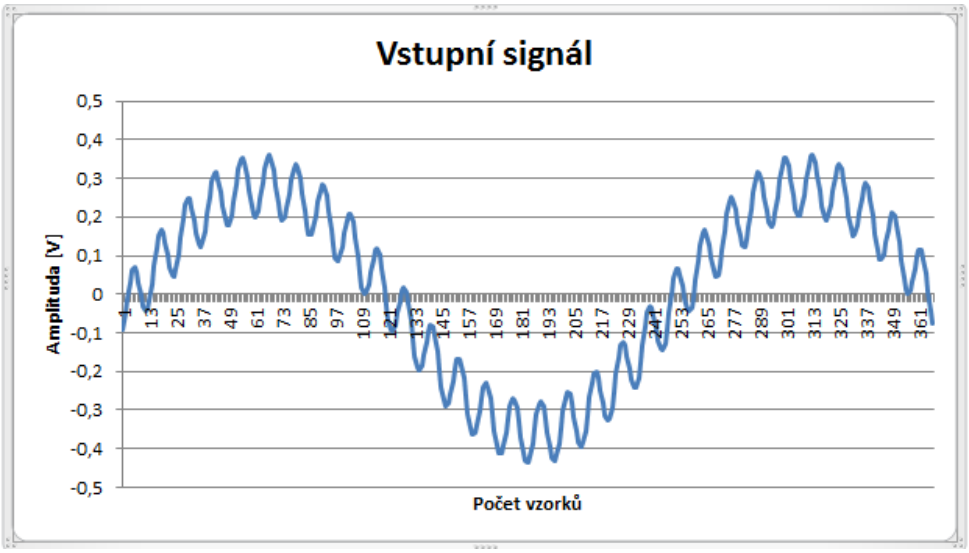
První signál:

Na obrázku 6.5 je zobrazen vstupní signál generovaný programem Waveforms. Signál je stejný jako při návrhu FIR filtru.



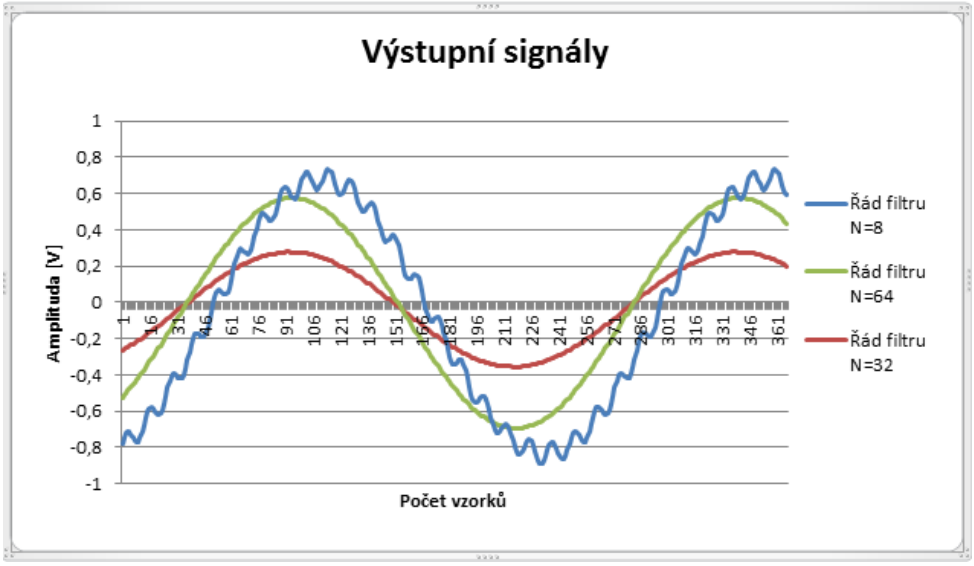
Obrázek 6.5: Vstupní signál vygenerovaný programem Waveforms

Do A/D převodníku se nahrává signál z programu Waveforms. Na následujícím obrázku je zobrazen vstupní signál $x = 1 \cdot sin(2 \cdot \pi \cdot t) + 0,25 \cdot sin(2 \cdot \pi \cdot 20 \cdot t)$ vykreslený v programu Microsoft Excel.



Obrázek 6.6: Vstupní signál v programu Microsoft Excel

Na obrázku 6.7 je zobrazen vyfiltrovaný signál pomocí metody váhových oken při řádu filtru N=8, N=32 a N=64

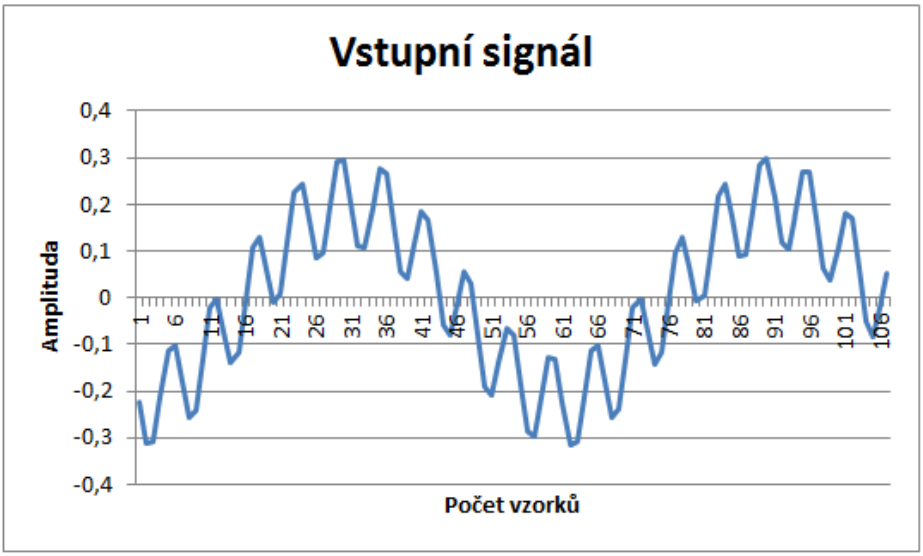


Obrázek 6.7: Vstupní signál v programu Microsoft Excel

Druhý signál:

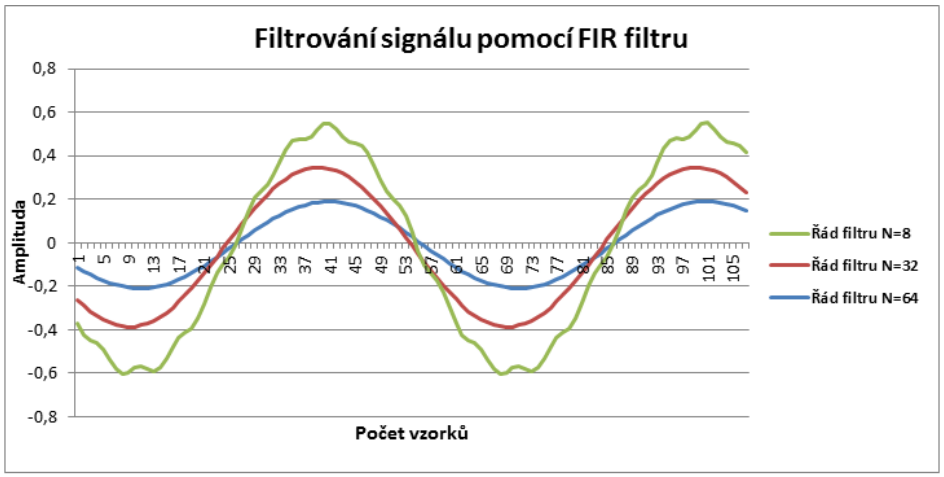
Signál byl generován pomocí programu Waveforms a pomocí signálového generátoru AnalogDiscovery byl tento signál přiveden na kit a přijímaný analogový signál byl pomocí A/D převodníku převeden na digitální. Signál byl filtrován pomocí FIR filtru, pomocí metody váhových oken, typ okna Hammingovo s využitím MMAU, řád filtru byl zvolen N=64, N=32 a N=8.

Na obrázku 6.8 je zobrazen vstupní signál $x = sin(2 \cdot \pi \cdot t \cdot 50) + 0.5 \cdot sin(2 \cdot \pi \cdot t \cdot 500)$



Obrázek 6.8: Vstupní signál

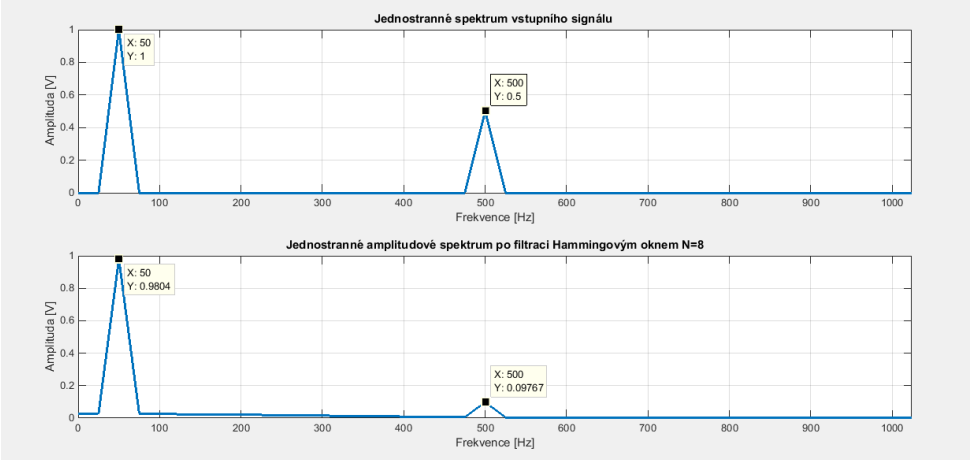
Na následujícím obrázku je vykreslen vyfiltrovaný signál pro řád filtru N=64 (modrý), N=32 (červený), N=8 (zelený).



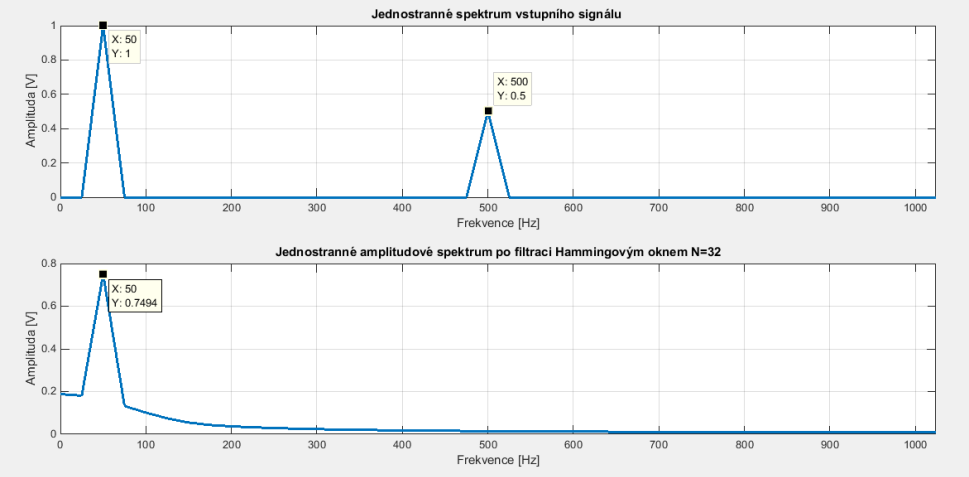
Obrázek 6.9: Výstupní signál pro řád filtru N=64, N=32 a N=8

Z obrázku lze vyčíst, že při řádu filtru N=8 se původní signál téměř vůbec nevyfiltroval. Při řádu filtru N=32 a N=64 je signál vyfiltrován.

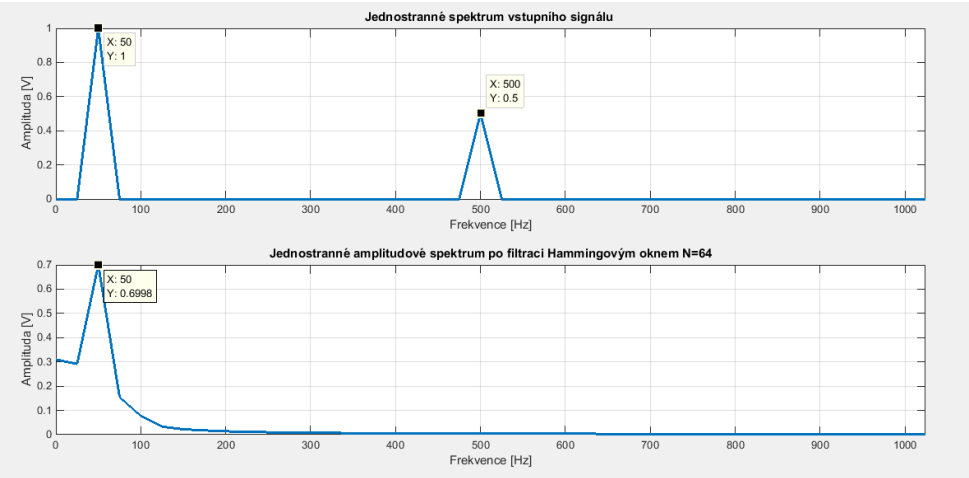
Na následujících obrázcích je zobrazeno jednostranné amplitudové spektrum pro řád filtru N=8, N=32 a N=64. Při filtrování signálu při použití řádu filtru N=8 lze ze spektra vidět, že signál není plně vyfiltrován. V signálu zůstal šum o amplitudě 0,097V. Při řádu filtru N=32 a N=64 je signál již vyfiltrován.



Obrázek 6.10: Jednostranné amplitudové spektrum pro řád filtru N=8

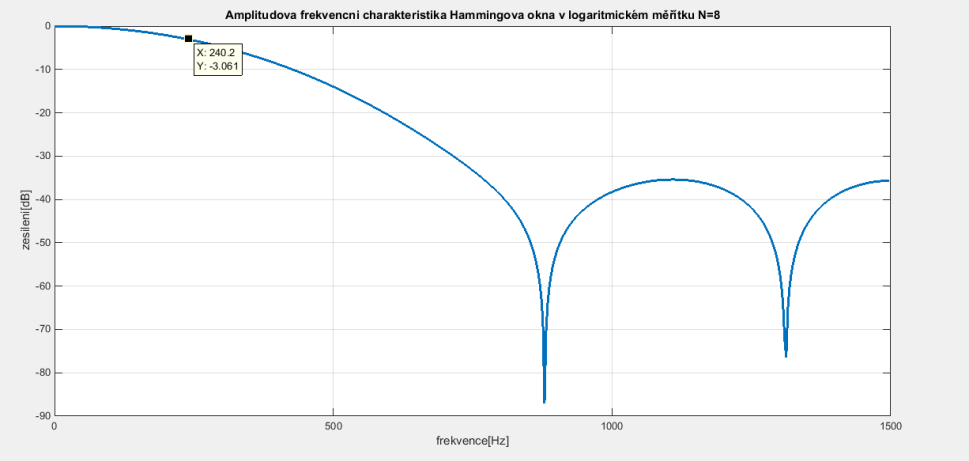


Obrázek 6.11: Jednostranné amplitudové spektrum pro řád filtru N=32

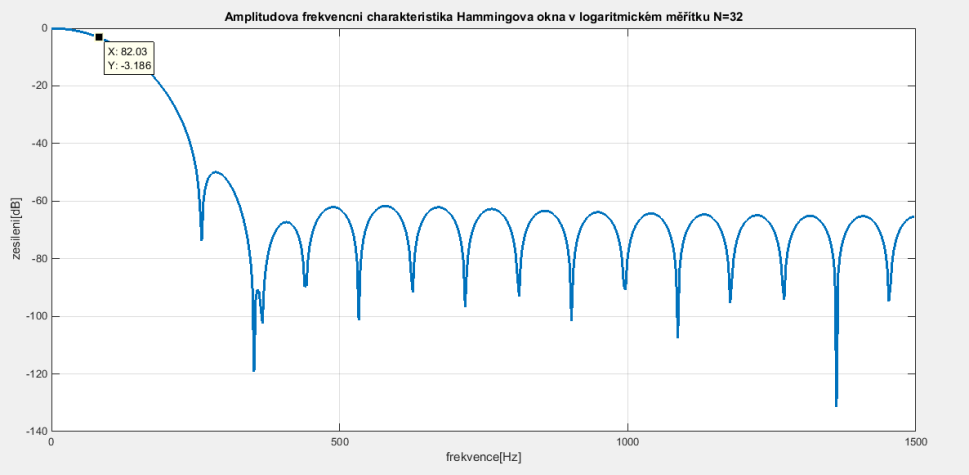


Obrázek 6.12: Jednostranné amplitudové spektrum pro řád filtru N=64

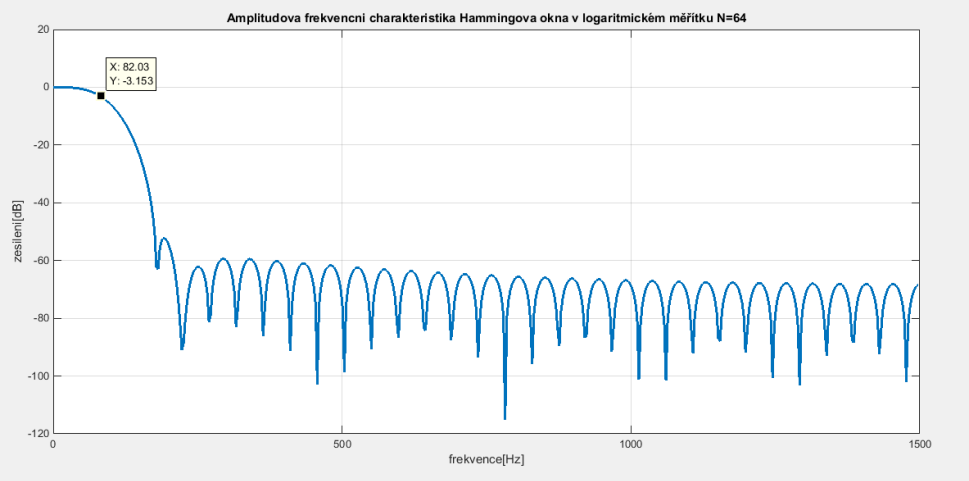
Na obrázcích 6.17, 6.18 a 6.19 je zobrazena amplitudová frekvenční charakteristika pro řád filtru N=8, N=32 a N=64. Při útlumu -3dB je u řádu filtru N=8 frekvence 240Hz, u řádu filtru N=32 82,03Hz a při řádu filtru N=64 je frekvence 82,03Hz. Filtr s řádem filtru N=64 má větší strmost než u řádu filtru N=32 a v propustném pásmu méně tlumí signál.



Obrázek 6.13: Amplitudová frekvenční charakteristika pro řád filtru N=8



Obrázek 6.14: Amplitudová frekvenční charakteristika pro řád filtru N=32



Obrázek 6.15: Amplitudová frekvenční charakteristika pro řád filtru N=64

6.4 Zhodnocení

Na základě těchto implementací se zkoumalo, kolik cyklů bude trvat procesoru než vyfiltruje signál pomocí IIR a FIR filtru bez použití MMAU, s použitím MMAU s návratovou hodnotou a s použitím MMAU bez návratové hodnoty. Pro tyto výpočty byla použita funkce

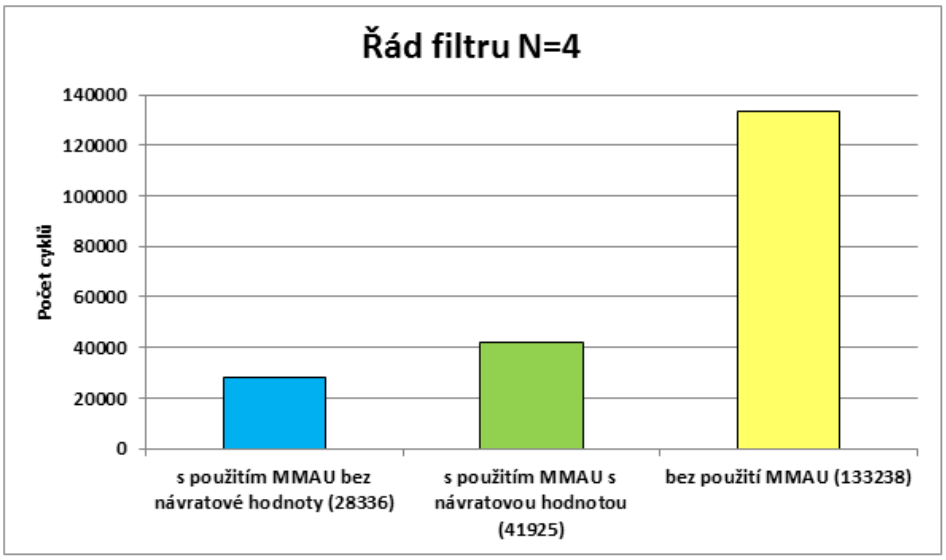
```
SYST_ClrCntrVal ();
```

Tato funkce vrací počet cyklů, které vykoná procesor, než se vykoná daná funkce. Počet vykonaných cyklů se dá lehce převést na sekundy podle následujícího vzorce

$$delka = \frac{1}{48000000} \cdot pocetcyklu.$$

IIR filtry

V grafu je znázorněno, kolik počtu cyklů vykoná procesor při filtrování signálu pomocí IIR filtru s použitím MMAU bez návratové hodnoty, s použitím MMAU s návratovou hodnotou a bez využití MMAU.



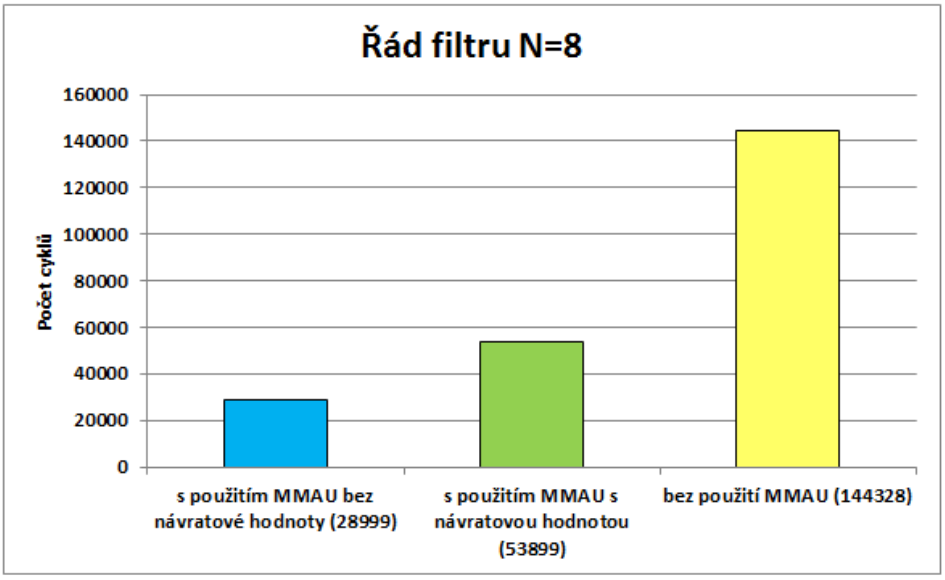
Obrázek 6.16: Graf IIR filtru

Při převodu na ms nejdéle trvalo vyfiltrování signálu bez použití MMAU a to 2,776 ms. Jako druhé v pořadí skončila funkce s použitím MMAU s návratovou hodnotou, která trvala 0,873 ms. Nejrychlejší bylo s použitím MMAU bez návratové hodnoty, které trvalo 0,59 ms.

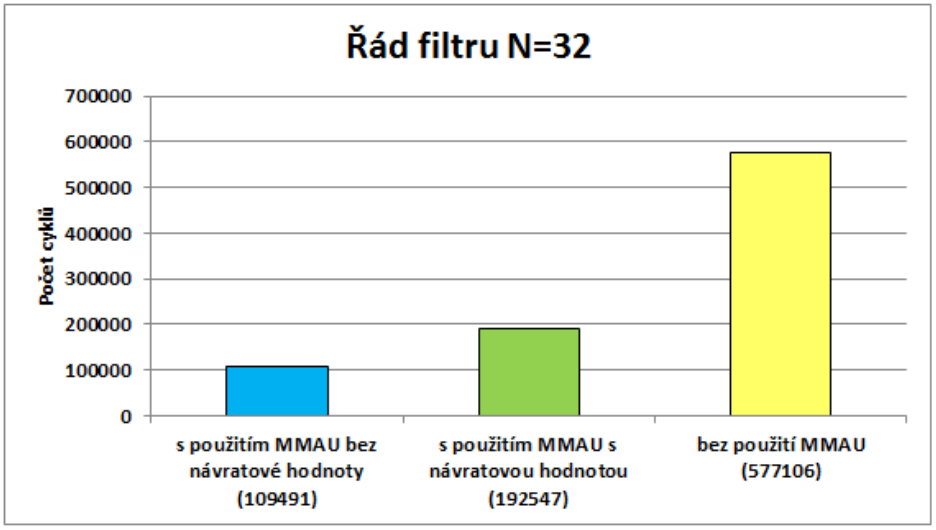
Srovnáním těchto časů vyplývá, že při filtrování pomocí IIR filtru bylo s použitím MMAU až několikanásobně rychlejší.

FIR filtry

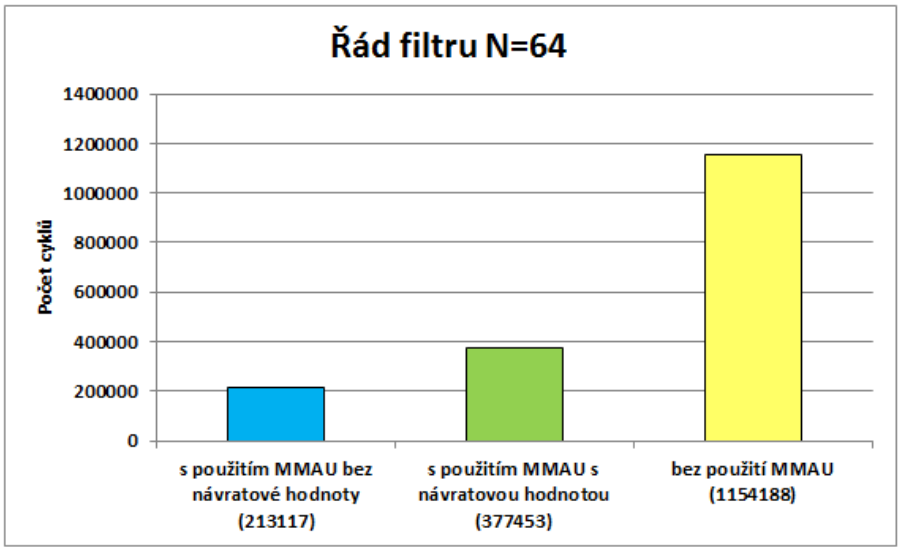
V následujících grafech je znázorněn počet cyklů, které vykonal procesor, než se vykonalo vyfiltrování prvního signálu pomocí FIR filtru při řádu filtru N=8, N=32 a N=64. Byly hodnoceny časy při použití MMAU bez návratové hodnoty, s použitím MMAU bez návratové hodnoty a bez využití MMAU.



Obrázek 6.17: Počet cyklů u řádu filtru N=8



Obrázek 6.18: Počet cyklů u řádu filtru N=32



Obrázek 6.19: Počet cyklů u řádu filtru N=64

Při řádu filtru N=8 se signál vyfiltroval nejrychleji při použití MMAU bez návratové hodnoty za 0,604ms, při použití MMAU s návratovou hodnotou trval výpočet 1,123ms
a bez použití MMAU filtrování trvalo 3,007ms.

Při řádu filtru N=32 se počet cyklů zvyšuje oproti řádu filtru N=8. Při použití MMAU bez návratové hodnoty výpočet trval 2,281ms, při použití MMAU s návratovou hodnotou bylo filtrování signálu téměř dvakrát delší než bez návratové hodnoty tedy 4,011ms a bez použití MMAU byl výpočet nejdelší 12,023ms.

Při řádu filtru N=64 byly výsledné časy pro vyfiltrování signálu následovné: při použití MMAU bez návratové hodnoty výpočet trval 4,44ms, při použití MMAU s návratovou hodnotou byla vykonaná filtrace za 7,863ms a bez použití MMAU byl signál vyfiltrován za 24,046ms.

Ve všech naměřených situacích byla nejrychlejší metoda s použitím MMAU bez návratové hodnoty. Při použití řádu filtru N=8 a N=32 byl nárůst času téměř čtyřnásobný. Při použití řádu filtru N=32 a N=64 byl nárůst délky filtrování téměř dvojnásobný. U filtrování signálu pomocí metody s použitím MMAU s návratovou hodnotou a bez použití MMAU byl výsledek podobný.

Se zvyšujícím se násobkem řádu filtru je délka filtrování násobně delší.

V této kapitole bylo čerpáno z [11].

Závěr

Cílem této práce bylo seznámení se s procesorem řady Kinetis M s jádrem ARM Cortex M0+ a s jeho funkcemi. Popis algoritmů na zpracování digitálního signálu a jejich implementace za použití procesoru ARM Cortex M0+.

FIR a IIR filtry byly nejprve navrženy v programu MATLAB. FIR filtry byly navrženy pomocí metody váhových oken, bylo použito Hammingovo a Obdélníkové okno, řády filtru byly vybrány N=32 a N=64. IIR filtry byly navrženy pomocí Butterworthovy aproximace, řád filtru byl zvolen N=4. V návrhu byly na daný vstupní signál vykresleny amplitudové frekvenční charakteristiky filtru, jednostranné amplitudové spektra před a po vyfiltrování, které ukazují, jestli filtr vyfiltroval veškerý šum, impulsní charakteristiky filtrů a signál po vyfiltrování.

FIR a IIR filtry byly implementovány v jazyku C, v programu IAR Embeeded Workbench na kitu TWR-KM34Z50M s procesorem ARM Cortex M0+ s využitím a bez využití matematického koprocesoru. Implementovány byly FIR i IIR filtry s použitím MMAU s návratovou hodnotou, s použitím MMAU bez návratové hodnoty a bez použití MMAU.

Za použití těchto metod byl následně změřen čas, jak dlouho trvalo procesoru vyfiltrování daného signálu pomocí algoritmů IIR a FIR filtrů. Je prokázáno zjištění, že nejrychleji trvalo vyfiltrování s použitím MMAU bez návratové hodnoty. Při řádu filtru N=8 trvalo vyfiltrování daného signálu 0,604ms, při řádu filtru N=32 2,281ms a při řádu filtru N=63 4,44ms. Výsledné časy byly až 5x rychlejší než bez použití MMAU a 2krát rychlejší než při použití MMAU s návratovou hodnotou. Je to dáno tím, že mezivýpočty zůstávají uloženy ve vnitřním akumulátoru MMAU.

Literatura

[1] Cortex-M Series, *ARM – The architecture for the Digtal world*,[online]. © 1995-2016 [cit. 2016-04-05]. Dostupné z: http://www.arm.com/products/processors/cortex-m/

[2] Cortex-M0+, Technical Reference Manual, *ARM – The architecture for the Digtal world*,[online]. © 2012 [cit. 2016-04-07]. Dostupné z: http://infocenter.arm.com/help/topic/com.arm.doc.ddi0484c/DDI0484C_cortex_m0p_r0p1_trm.pdf

[3] Cortex-M0+ Processor, *ARM – The architecture for the Digtal world*,[online]. © 1995-2016 [cit. 2016-04-07]. Dostupné z: http://www.arm.com/products/processors/cortex-m/cortex-m0plus.php

[4] Cortex-M3 Processor, *ARM – The architecture for the Digtal world*,[online]. © 1995-2016 [cit. 2016-04-10]. Dostupné z: http://www.arm.com/products/processors/cortex-m/cortex-m3.php

[5] KM Family Reference Manual, *NXP* ,[online]. 10/2013 [cit. 2016-04-17]. Dostupné z: http://www.nxp.com/files/microcontrollers/doc/ref_manual/MKMxxZxxCxx5RM.pdf

[6] VLČEK Jiří , *D/A a A/D převodníky*,[2009] [cit. 2016-04-20]. Dostupné z: http://elektro.tzb-info.cz/102-digitalni-ucebnice

[7] Analog Digital interface Integrated Circuits, *Free Book Centre*, © 2016-2017 [cit. 2016-03-21]. Dostupné z: http://www.freebookcentre.net/electronics-ebooks-download/Analog-Digital-interface-Integrated-Circuits.html

[8] VÍCH, Robert, Zdeněk Smékal. *Číslicové filtry.* Praha: Nakladatelství Academia, 2000. ISBN 80-200-0761-X.

[9] DAVÍDEK, Vratislav, Miloš Laipert, Miroslav Vlček. *Analogové číslicové filtry.* Praha: Nakladatelství ČVUT, 2006. ISBN 80-01-03026-1.

[10] SMÉKAL, Zdeněk, Petr Sysel. *Číslicové filtry.* 2014. [cit. 2016-04-22]. Dostupné z: https://www.researchgate.net/publication/47091716_Cislicove_filtry

[11] Exploring the 64-bit Memory Mapped Arithmetic Unit. *NXP* [online]. © 2015 [cit. 2016-04-01]. Dostupné z: http://cache.nxp.com/files/32bit/doc/app_note/AN5189.pdf?fsrch=1&sr=1&pageNum=1

Přílohy

Zdrojové kódy IIR a FIR filtrů a zdrojové kódy v programu Matlab jsou uloženy na přiloženém nosiči CD ROM. Zdrojové kódy jsou uloženy v souboru s názvem KM256SWDRV_R4_0_0, v podsložce build je spustitelný soubor projekt.eww a v podsložce src jsou uloženy použité knihovny. Zdrojové kódy v programu Matlab jsou uloženy v souborech IIR_filtry.m, FIR1_filtry.m, FIR2_filtry.m